Universidad
Carlos III de Madrid

Departamento de Informática

Trabajo Fin de Grado

Teleoperación del robot NAO mediante dispositivos móviles Android

Autor: Juan Domingo Gálvez Cobo

Director: Moisés Martínez Muñoz
Co-Director: Ezequiel Antonio Quintero Barrios

Título: Teleoperación del robot NAO mediante dispositivos móviles Android
Autor: Juan Domingo Gálvez Cobo
Director: Moisés Martínez Muñoz
Co-Director: Ezequiel Antonio Quintero Barrios

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día __ de _____de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Agradecer en primer lugar la ayuda prestada por mi tutor, Moisés Martínez Muñoz, y por el co-director, Ezequiel Antonio Quintero Barrios, a las aclaraciones a mis numerosas preguntas llenas de ganas de aprender.

Por supuesto, gracias a mis padres, por el aliento y el incondicional cariño que me han brindado en todo momento, y los valores que han sabido inculcarme para hacer de mi un hombre de provecho. A mi hermana, que aunque siempre hemos sido polos opuestos, ahora parece que empezamos a congeniar.

A mis “titos”, “titas”, “primos” y “primas” por esos grandes ratos de cariño, buena compañía y comilonas, los cuales me han hecho sentirme en Madrid como en casa.

Gracias a mis amigos y compañeros de clase, por su ayuda y apoyo, y porque sin ellos, este camino recorrido no hubiera sido igual.

Resumen

En este trabajo se presenta el diseño de un sistema de teleoperación para un robot humanoide mediante el uso de dispositivos móviles que utilizan el sistema operativo Android. Los objetivos principales de este trabajo consisten en la creación de una aplicación en Android que permita teleoperar el robot NAO, mediante una conexión inalámbrica. Así como el desarrollo de un sistema de comunicaciones que permita al robot enviar y recibir mensajes de la aplicación móvil e interpretarlos. Para el desarrollo de este trabajo se ha utilizado el sistema operativo ROS, debido a que existen un gran número de paquetes orientados al control de robots, el entorno de desarrollo Eclipse, para facilitar la programación tanto en C++ como en Android, el software de programación y simulación Choregraphe y NAOqi para realizar pruebas del sistema, y el robot NAO, en el que se ha comprobado el correcto funcionamiento del sistema desarrollado.

Palabras clave: Sistema de comunicaciones, dispositivos móviles, Android, robot humanoide, NAO, teleoperación, ROS, Eclipse, C++, programación, simulación, Choregraphe, NAOqi.

Abstract

This document shows the design of a teleoperation system for a humanoid robot through mobile devices which use Android Operating System. The main objectives in this work include the creation of an Android application that allows to control the NAO robot through a wireless connection. As well as the development of a communication system that allows the robot to send and receive messages from the mobile application and interpret them. For the development of this work, it has used the operating system ROS, which provides a large number of packages designed to control robots, the Eclipse development environment to make easy programming in both C++ and Android, the programming and simulation software NAOqi and Choregraphe for testing the system, and the NAO robot, in which has been tested the correct operation of this teleoperation system.

Keywords: Communication system, mobile devices, Android, humanoid robot, NAO, teleoperation, ROS, Eclipse, C + +, programming, simulation, Choregraphe, NAOqi.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se presenta una breve introducción del trabajo realizado, la motivación por la cual ha sido realizado, los objetivos que se definieron inicialmente en su desarrollo y un breve resumen de la estructura del documento.

1.1 Introducción

La robótica es la ciencia o rama de la tecnología mediante la cual se diseñan y construyen agentes físicos y sistemas capaces de realizar tareas de forma automática y en algunos casos de forma más eficiente que los seres humanos. Estos dispositivos son desarrollados para realizar tareas que no pueden ser llevadas a cabo por los seres humanos debido a la complejidad de las mismas y/o a su peligrosidad. Los avances realizados en los últimos años han permitido acercar los agentes físicos y los sistemas automáticos a la vida cotidiana de las personas, de forma que es necesario crear herramientas sencillas que permiten el control de estos dispositivos.

Uno de los principales problemas junto con la construcción de los agentes físicos, es el diseño e implementación de los sistemas de control que gobiernan sus comportamientos y la interacción con el entorno. Actualmente, es posible crear dos tipos de sistemas de control, los automáticos y los manuales o teleoperados. Los primeros, son aquellos que hacen uso de la inteligencia artificial para “pensar”, evaluar y realizar acciones de acuerdo a ciertos principios, satisfaciendo algún objetivo concreto. Y los

manuales o teleoperados, que ofrecen interfaces mediante las cuales el ser humano controla directamente el agente físico y puede llevar a cabo tareas para las que este ha sido diseñado.

Los sistemas teleoperados pueden definirse como el conjunto de *hardware* y *software* requerido para operar una máquina o dispositivo a una determinada distancia [1]. Requieren de una interfaz adecuada para la interacción hombre-máquina. Habitualmente consisten en dos agentes físicos, o un agente físico y un dispositivo de control, que están conectados de tal manera que el operador puede controlar el dispositivo principal (maestro), por medio del cual se generan instrucciones que se envían a distancia al agente físico (esclavo). Son cada vez más comunes en diversas áreas:

- Medicina [2], para realizar tareas complejas en microcirugía, ya que se requiere una gran precisión.

- Numerosas industrias, como la automovilística [3] o la nuclear [4], para eliminar o disminuir el riesgo para los seres humanos. Como en el caso de la radioactividad producida por ciertos materiales al ser manipulados.

- Labores de rescate [5], en catástrofes naturales como terremotos, tsunamis y huracanes. Para trabajar en zonas de difícil acceso para los humanos.

- Asistencia o tele-asistencia [6], a personas con ciertas discapacidades o dependencias.

- Movilidad en situaciones hostiles [7], evitando el peligro al que pueden verse sometidos los seres humanos, como en una guerra.

Aparte de los entornos mencionados, la teleoperación puede ser aplicada en otros ámbitos, en todos ellos con un propósito común, facilitar las tareas que vayan a ser realizadas. Para realizar estos trabajos se suelen emplear robots móviles con motores o servos, y dependiendo de cada caso, con distintas formas (vehículos, humanoides, zoomorfos,…) y tamaños, con el fin de adaptarse a las tareas para las que se van a destinar y el entorno en el cual van a interactuar.

En este trabajo, se ha desarrollado un sistema de teleoperación para el robot humanoide NAO, mediante la utilización de dispositivos que utilicen el Sistema Operativo Android. NAO es un robot humanoide de 57 cm de estatura desarrollado por la empresa francesa Aldeberan Robotics. Consta de varios tipos de sensores, táctiles, un sonar, cámaras, etc. Android es el sistema operativo basado en Java, utilizado por numerosos dispositivos móviles en la actualidad. Además, existe un número elevado de dispositivos de bajo coste que lo utilizan, haciéndolo accesible para todo el mundo.

1.2 Motivación

En los últimos años han aparecido un amplio número de agentes físicos, los cuales son capaces de llevar a cabo tareas más complejas. Así como nuevos dispositivos móviles más potentes, y con mejores características. Su expansión como plataformas para el desarrollo y su bajo coste, hacen posible el diseño y la creación de aplicaciones amigables para usarlas como sistemas de control en los robots actuales. Haciendo a su vez más accesible estos recursos a los centros universitarios y de investigación, llegando así a personas con menos recursos. A continuación se describen de forma detallada, las distintas motivaciones que me han llevado a la realización del trabajo:

- Actualmente existen un amplio número de tareas que pueden ser realizadas mediante robots teleoperados. Algunas de ellas para mejorar las técnicas utilizadas por los humanos (como en operaciones de microcirugía) o mejorar la calidad de vida de una persona (como robots de asistencia personal). En otros casos, debido al riesgo que la tarea conlleva para una persona (tareas de rescate, manipulación de elementos perjudiciales para el ser humano, extracción de minerales, o exploración de minas o fondos marinos).

- La reciente aparición de los nuevos sistemas operativos móviles, como Android o iOS, ofrecen una plataforma de desarrollo que permite la creación de aplicaciones que se puedan usar para la teleoperación de robots.

- La curiosidad personal de aprender cómo funciona el sistema de teleoperación de un robot, cómo poder controlarlo y ver las posibilidades que tiene la teleoperación utilizando dispositivos móviles actuales, accesibles para cualquier persona. Así como programar aplicaciones en Android y manejar el sistema operativo ROS.

1.3 Objetivos

El objetivo principal de este trabajo, consiste en el diseño y desarrollo de un sistema de teleoperación para el robot NAO, mediante la utilización de un dispositivo móvil táctil que utilice el sistema operativo Android. A continuación se describen de forma detallada los diferentes objetivos parciales extraídos del objetivo principal de este trabajo:

1. Estudio y análisis de ROS (*Robot Operating System*). Debido a que ha sido la herramienta elegida para el desarrollo del sistema de teleoperación del robot. Por lo que es necesario un estudio de las distintas funcionalidades que este ofrece, las cuales influirán en el diseño del sistema de teleoperación.

2. Estudio y análisis del sistema operativo Android. Debido a que ha sido el sistema operativo elegido para desarrollar la interfaz de teleoperación para el robot. Esto implica el estudio de las diferentes formas de desarrollar la aplicación, y la selección de los métodos y librerías más adecuados para definir el diseño de la interfaz y facilitar el proceso de desarrollo.

3. Diseño e implementación del sistema de teleoperación.

    - Estudio y análisis de trabajos similares. Con el fin de desarrollar un sistema de teleoperación que se adapte a los objetivos de este trabajo, es necesario conocer los elementos básicos y el funcionamiento de este tipo de sistema. Para ello se han consultado diversas fuentes (libros, revistas, internet), asimilando el funcionamiento de sistemas de control similares que utilizan ROS y Android, o sistemas análogos.

    - Diseño del módulo de control. Establecer las características principales que el módulo va a tener, contando con cada una de las posibilidades que el *framework* seleccionado ofrece.

    - Implementación del módulo de control. Después de establecer los elementos que van a conformar la arquitectura del módulo en el diseño, será necesario implementar cada una de estas partes.

    - Diseño del módulo de teleoperación. Establecer los elementos principales que la aplicación va a tener, observando en cada momento como puede ser más amigable para el posterior uso por parte de los usuarios, y teniendo en cuenta los problemas que puedan surgir debido a la interfaz.

    - Implementación del módulo de teleoperación. Tras diseñar la aplicación, se procederá a desarrollarla teniendo en cuenta los factores anteriormente mencionados.

4. Experimentación y evaluación del sistema desarrollado.

    - Diseño de experimentos. Definición de un conjunto de tareas que permitan utilizar las distintas funcionalidades que han sido desarrolladas en el sistema de teleoperación.

    - Realización de experimentos. Selección de un conjunto de usuarios que ejecuten las distintas tareas propuestas previamente. Con el fin de comprobar las dificultades que aparecen al utilizar la herramienta desarrollada, así como los diferentes problemas derivados del proceso de control del robot NAO.

    - Análisis de resultados. Evaluación y análisis de los resultados obtenidos en las tareas realizadas en la experimentación, para comprobar si el sistema desarrollado se adapta a las funcionalidades definidas inicialmente. Así como obtener información para el desarrollo de posteriores mejoras en el sistema.

5. Desarrollo de la documentación. Mediante la cual se describe el funcionamiento del sistema que ha sido desarrollado. Así como el proceso que se ha seguido para implementarlo.

1.4 Estructura del documento

Este documento está dividido en 6 capítulos y 3 anexos. A continuación se realiza una descripción del contenido de cada uno de ellos:

- En el capítulo 1 se presenta la introducción al trabajo, la motivación que me ha impulsado en su elaboración, los objetivos que van a cubrirse durante su desarrollo y un breve resumen de la estructura del documento.

- En el capítulo 2 se presenta el marco teórico, donde se realiza una breve descripción de la evolución de la robótica, y de las diferentes técnicas de control existentes para gobernar un robot o agente físico. Realizando una descripción detallada de las técnicas de teleoperación, las cuales están estrechamente relacionadas con este proyecto.

- En el capítulo 3 se realiza una descripción detallada del sistema de teleoperación que ha sido desarrollado. En él se presenta el análisis y diseño de la aplicación, así como el proceso que se ha seguido para llegar al sistema final.

- En el capítulo 4 se describen las pruebas que han sido llevadas a cabo para comprobar el correcto funcionamiento del sistema. Este proceso se realizó mediante evaluaciones individuales por parte de usuarios. Estas evaluaciones estaban constituidas por un conjunto de pruebas de control, usabilidad, eficiencia y eficacia del sistema de teleoperación.

- En el capítulo 5 se detalla la información referente a la planificación del desarrollo de este trabajo, los medios empleados, y los costes totales.

- En el capítulo 6 se presentan las conclusiones obtenidas, junto con la revisión de los objetivos propuestos y las posibles líneas futuras que podrían llevarse a cabo sobre la herramienta desarrollada.

- En al anexo A se describe el proceso de instalación y configuración del entorno adecuado para poder realizar el trabajo que se ha llevado a cabo.

- En el anexo B se muestran los diagramas de clases del sistema de teleoperación desarrollado. Realizando una breve descripción de los mismos.

- En el anexo C se describe el manual de uso de la aplicación creada en Android.

Capítulo 2

Estado del arte

El origen de la palabra **robot** se remonta a 1921, cuando Karel Câpek (1890-1938) la usó por primera vez en su novela Rossum's Universal Robot. En el libro se presenta un sociedad ficticia en la cual los robot creados por la corporación Rossum, servían a los humanos realizando trabajos físicos. Hasta que llega el día en el que se sublevan, acabando con todo ser humano menos con uno de sus creadores. Con el propósito de que les mostrara cómo reproducirse y perpetuarse en el tiempo. El término proviene de la palabra eslava *robota*, que significa trabajo que se realiza forzadamente.

Cabe destacar a Isaac Asimov (1920-1992) escritor y bioquímico, creador del concepto "robótica". Escribió numerosas obras donde hablaba de los robots, incorporando en algunas de ellas ("Yo, Robot" y Robots e Imperio) las 3 leyes por las que se deberían regir los sistemas de control de los robots, de forma que nunca sucediera lo que nos cuenta en su historia Karel Câpek. Muchos de sus relatos han influido en los investigadores a la hora de crear los robots modernos.

2.1 Historia de la robótica

Desde la antigua Grecia, los seres humanos han intentado simular las acciones realizadas por los seres vivos, mediante la creación de sistemas automáticos denominados *automato*, derivando en los que actualmente conocemos como autómatas o robots.

En el 400 a.c. en Alejandría se crearon algunos de los primeros autómatas conocidos. El teatro de autómatas y los Pájaros, presentados en la Figura 1 y la Figura 2 respectivamente, ambos de carácter recreativo, construidos por Herón de Alejandría, inventor y matemático. Posteriormente los árabes utilizaron los conocimientos de Grecia y los llevaron a la práctica por medio de la construcción de sistemas de abastecimiento de agua.

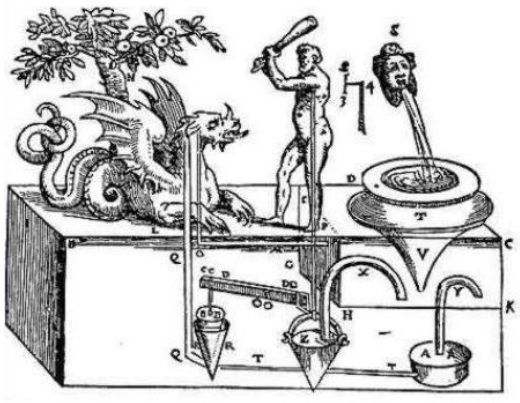

Figura 1. Teatro de autómatas.

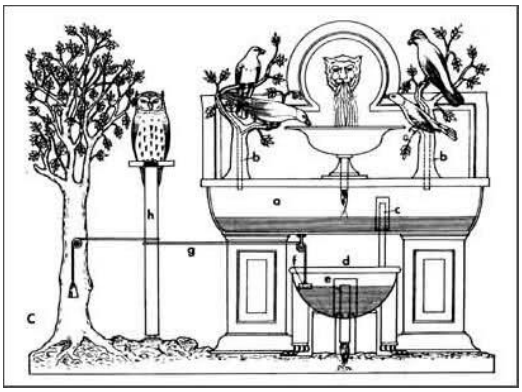

Figura 2. Pájaros de Herón.

El Gallo de Estrasburgo, presentado en la Figura 3, es el autómata más antiguo (1352) que se ha conservado. Situado en el reloj de la catedral de la ciudad de Estrasburgo, imitaba el comportamiento de un gallo moviendo las alas y el pico al dar las horas. Otros de los más conocidos que se crearon en siglos posteriores son el León mecánico y el Pato de Vaucanson, de Leonardo Da Vinci y del relojero suizo Pierre Jaquet Droz respectivamente.

Figura 3. Gallo de Estrasburgo.

Pero no fue hasta la revolución industrial cuando aparecieron los primeros autómatas que sustituyeron a los seres humanos en la realización de tareas complejas o de gran precisión. Como la hiladora mecánica de Cromptron (1779) presentada en la Figura 4, y los telares mecánicos de Cartwirght (1785) y de Jacquard (1801).

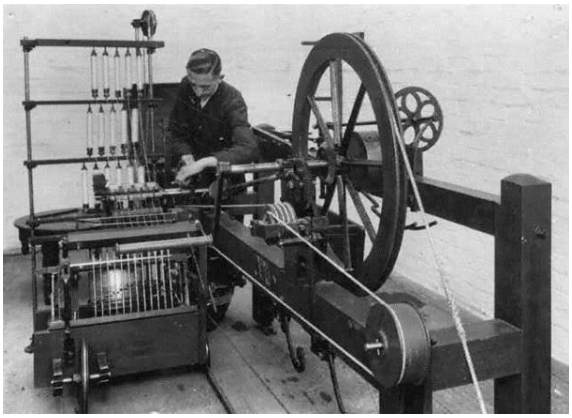

Figura 4. Hiladora mecánica de Cromptron.

Sin embargo, no fue hasta la mitad del siglo XX cuando se produjo una verdadera revolución en el diseño y desarrollo de autómatas, gracias a la invención de los transistores y los circuitos integrados. En 1948 Grey Walter construye "Las tortugas" (Figura 5), que únicamente hacían dos cosas: esquivar obstáculos y volver al punto de recarga antes de que se les agotaran las baterías. George Charles Devol, inventor estadounidense del primer robot industrial programable llamado Unimate (1954) que se instaló en una cadena de montaje de General Motors en Ewing (Nueva Jersey) en 1961, estableció las bases de la robótica industrial moderna.

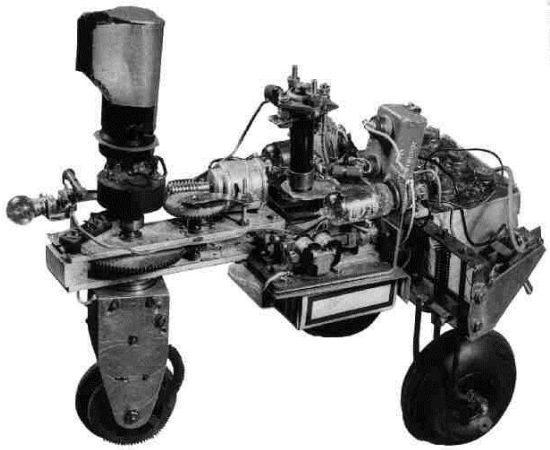

Figura 5. Tortuga de Grey Walter.

Otro de los importantes pasos hacia la robótica moderna se dio cuando se empezaron a utilizar programas de ordenador para controlar los movimientos de las máquinas, donde antes los realizaba un operador humano. En la década de los 60 se creó el robot Shakey (Figura 6), uno de los primeros robots controlados por computador mediante el uso de Planificación Automática, en el Centro de Inteligencia Artificial del Instituto de

investigación de Standford (SRI International). Fue el primer proyecto que fusionó el razonamiento lógico y la acción física.

Figura 6. Robot Shakey.

A partir de la década de los 70, comenzaron a aparecer una gran variedad de robots complejos, debido a la creación de dispositivos eléctricos y mecánicos mejores y más pequeños.

- En 1975 se creó PUMA, un brazo robot industrial dedicado a tareas de montaje, desarrollado por Victor Scheinman en la empresa Unimation.

- En 1977 la Voyager 1 y 2 son lanzadas al espacio, sondas compuestas de varios sensores y cámaras para adquisición de datos.

- En 1988, LEGO presentó su primer producto (LEGO TC Logo). Con este, los alumnos de las escuelas podían programar el comportamiento de sus construcciones de LEGO.

- En 1989 Rodney A. Brooks creó en el MIT el robot Genghis (Figura 7), un robot hexápodo caminante, de anatomía similar a la de un insecto.

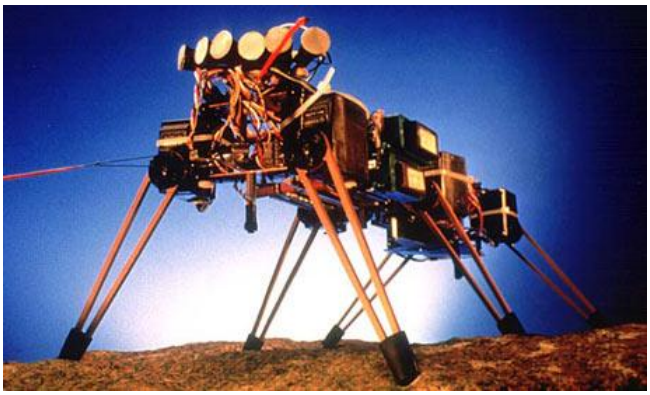

Figura 7. Robot Genghis.

- En 1993 y 1994 respectivamente, fueron creados Dante y Dante II, robots de ocho patas diseñados para escalar pendientes y hacer rappel por escarpados muros para recoger datos en terrenos peligrosos para el ser humano.

- El robot humanoide de Honda, P2, fue mostrado en 1996. Tenía la capacidad de andar y subir escaleras, con unos movimientos muy similares a los del ser humano.

- En 1998, LEGO lanzó el kit de construcción MindStorms, con el que se pueden construir y programar robots sencillos.

- En el 2000, Honda mostró el resultado más avanzado de su proyecto humanoide, llamado ASIMO (Figura 8). Este robot es capaz de correr, caminar, comunicarse con los seres humanos, reconocimiento facial y de voz, e interactuar con el medio en el que se encuentra.

Figura 8. Robot humanoide ASIMO.

- En 2004, aterrizaron en Marte Spirit (Figura 9) y Opportunity, los Rovers autónomos enviados por la NASA para la exploración del terreno marciano.

Figura 9. Rover Spirit.

- En 2007, Toyota presentó un robot humanoide capaz de interpretar piezas musicales de violín, imitando los gestos humanos necesarios para producir sonidos como el "vibrato".

- En 2008, NAO, robot autónomo programable, fue lanzado a los participantes de la RoboCup, y posteriormente al mercado académico y empresarial.

Figura 10. Nao primera generación.

- En 2010, Willow Garaje vende PR2, su plataforma robótica abierta de investigación y desarrollo.

- En 2011 la NASA trasladó a Robonaut (Figura 11), robot humanoide autónomo, a la estación espacial internacional. Es el primer robot de este tipo en ponerse en órbita. Su función principal es la de ayudar en tareas peligrosas para el ser humano.

Figura 11. Robot Robonaut de la NASA.

- El último hito en la robótica espacial es el rover Curiosity, que aterrizó en Marte el 6 de agosto de 2012. Enviado para tomar muestras e investigar la capacidad del planeta para alojar vida.

Figura 12. Rover Curiosity.

2.2 Teleoperación

La teleoperación es un área de la robótica que consiste en el desarrollo de interfaces que permitan el control de diferentes robots de forma remota por un ser humano. Por lo que un sistema de teleoperación permite controlar el movimiento y/o la fuerza ejercida por un robot (esclavo) que está situado en una localización diferente a la del operador por medio del manejo de un dispositivo de control (maestro) que se sitúa en el mismo lugar que el operador. El maestro por medio de una interfaz, se comunica con el esclavo a través de un canal, ya sea físico, como motores y servos, o de la transmisión de los datos del movimiento por medio de algún canal (Wifi, infrarrojos, *bluetooth*,…). La Figura 13 muestra un esquema de la teleoperación maestro esclavo.

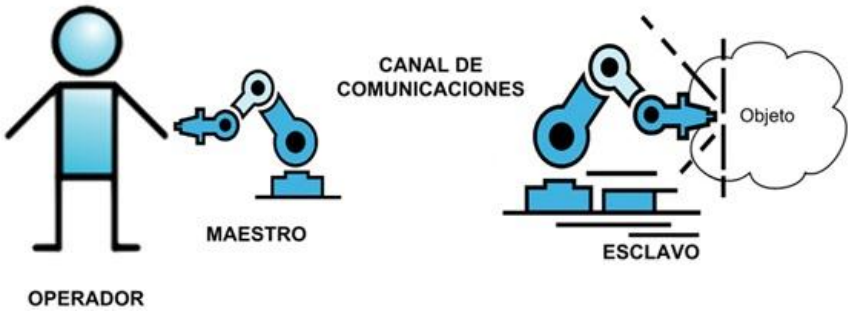

Figura 13. Esquema teleoperación maestro esclavo.

El primer sistema teleoperado fue desarrollado por un grupo de investigadores encabezados por Raymond Goertz en 1948. Desarrollaron el primer manipulador mecánico maestro/esclavo teleoperado para poder manejar materiales radioactivos, denominado M1 (Figura 14). Este sistema permitía reproducir exactamente el movimiento que se realizaba por parte del operador, en el lado opuesto, evitando así los peligros que conllevaban manipular productos radioactivos. Años después se desarrolló una versión en la que se utilizaban motores eléctricos y servocontroles, denominada E1.

Figura 14. Manipulador M1.

El primer sector en el que se comenzó a utilizar la teleoperación fue en el de la energía nuclear, pero a partir de 1960, con la aparición de nuevos sensores (cámaras, sonars, etc.) y nuevos actuadores, los sistemas teleoperados se empezaron a utilizar en otras áreas:

- Investigación submarina: recogida de información de los fondos oceánicos, inspección para la posible extracción de minerales, o construcción de instalaciones.

- Aero-espacial: recogida de muestras en otros cuerpos celestes, misiones no tripuladas, mantenimiento de estaciones espaciales o satélites. Se reducen los riesgos para los astronautas y se reduce el costo derivado del equipo. Un ejemplo de misiones no tripuladas son la Voyager 1 (Figura 15) y 2, lanzadas en 1977.

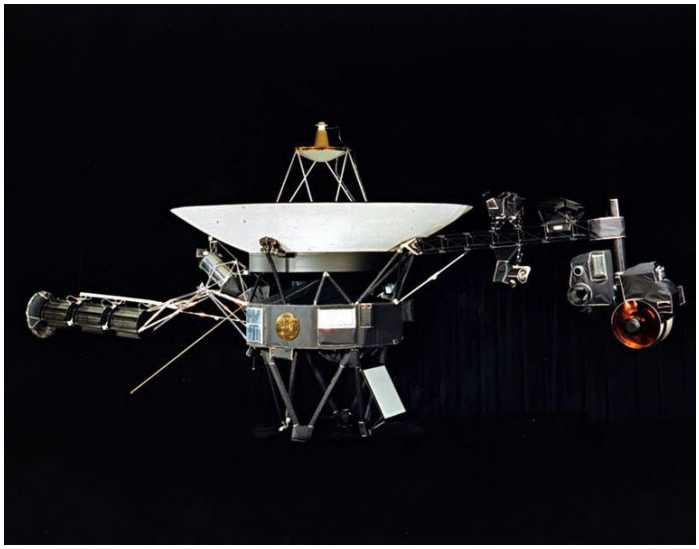

Figura 15. Voyager 1.

- Militar: control de vehículos aéreos no tripulados, como el RQ-3 DarkStar construido en 1996 (Figura 16), para detección de enemigos, vigilancia,…, sin el riesgo físico que conllevaría si el controlador estuviera en el vehículo. Vehículos terrestres trabajando en la desactivación de explosivos, vigilancia u otro tipo de labor para garantizar la seguridad a las personas.

Figura 16. RQ-3 DarkStar.

- Médicas: operaciones que requieren mucha precisión, o que por la escala a la que se realizan se necesitan herramientas muy pequeñas y aumentos en la visión. Para ello se utilizan equipos de microcirugía teleoperada como Raven, en 2012 (Figura 17).

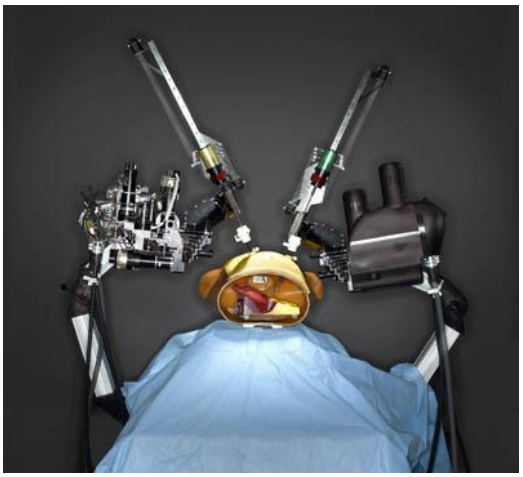

Figura 17. Raven.

- Otros sectores: mantenimiento de cableado eléctrico, labores de inspección, exploración y rescate en entornos de difícil actuación, cuidado personal, micro robots,…

2.2.1 Métodos de teleoperación

Para analizar los métodos de teleoperación, se van a tener en cuenta dos principios importantes: el tipo de control que ejerce el usuario sobre el robot, y la información que este recibe de vuelta. A partir de estos factores se definen los distintos métodos.

2.2.1.1 Control bilateral

Son aquellos métodos para los que existe algún tipo de realimentación de vuelta al operador, proveniente de las fuerzas que recibe el esclavo al realizar alguna tarea. Se llevan a cabo para poder impedir que el robot esclavo se dañe o para no ejercer fuerzas excesivas, sobre ciertos materiales. Tras realizar el operador un movimiento en el lado maestro, se obtiene su posición y esta es transmitida al esclavo, adaptándose a la del maestro. En el lado del esclavo se obtiene a su vez la posición y es enviada al maestro. La reflexión de las fuerzas se hace visible cuando hay diferencias entre la ubicación del maestro y el esclavo. El operador controla los ángulos de movimiento del robot totalmente [8].

El control que utiliza HIRO III (Figura 18), desarrollado por científicos de la Universidad Gifu y el Laboratorio Mouri en Japón, es bilateral, a través de las yemas de los dedos transmite sensaciones realistas de tacto. Consta de un brazo robótico con una

mano en la que se sujetan los dedos del operador, así como una pantalla tridimensional y unas gafas para poder verla.

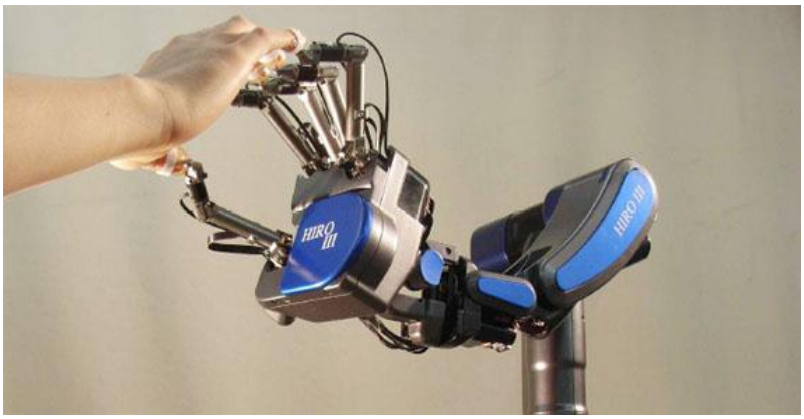

Figura 18. HIRO III, Simula las sensaciones táctiles que tendría el operador.

Telesar V es un robot humanoide desarrollado por la universidad de Keio (Japón), en el que también se utilizan dispositivos para las manos, introduciendo al usuario por medio de un sistema de visión en un mundo de realidad virtual, percibiendo directamente lo que el robot ve, y por medio de los dispositivos de las manos, sintiendo lo que el robot siente, ya que tiene un sistema de retroalimentación hacia el usuario. Por medio de servo motores, el usuario siente en la yema de sus dedos, lo que el robot está tocando, Figura 19.

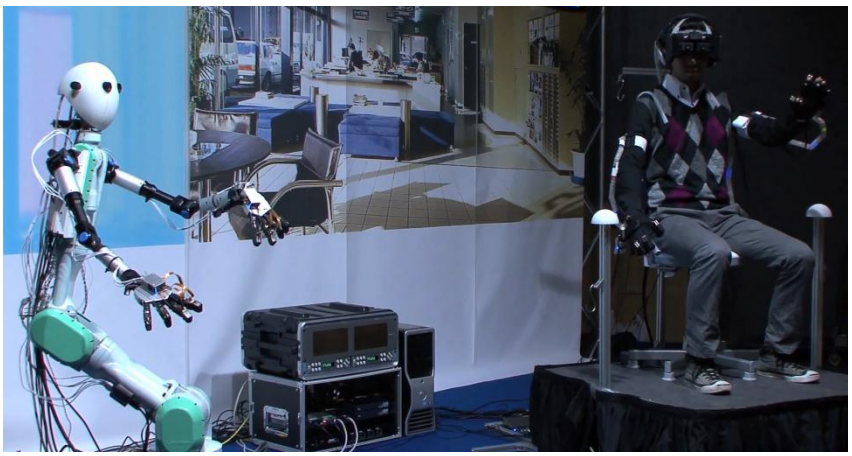

Figura 19. Robot teleoperado Telesar V.

2.2.1.2 Control supervisado y coordinado

Los sistemas de control supervisado y coordinado son aquellos que ejecutan acciones de más alto nivel que en el caso anterior, ejerciendo el operador una acción de supervisión. Existe entre maestro y esclavo un canal de comunicaciones por el que se mandan las instrucciones y la respuesta a esas instrucciones. El maestro realiza el control de una manera indirecta, ya que existen retardos debido al canal por el que se comunican ambos. Actualmente es el sistema de teleoperación más extendido y sobre el cual se están realizando un mayor número de investigaciones. Permite la realización de pruebas con

modelos 3D. En este tipo de control se basa este trabajo, donde por medio de una red Wifi se comunican el dispositivo Android de control y el robot NAO.

BigDog (Figura 20) es un robot cuadrúpedo creado en 2005 por las compañías Boston Dynamics, Foster-Miller, el Laboratorio de Propulsión a Chorro de la Nasa y la Concord Field Station de la Universidad de Harvard, para uso militar. El operador manda las instrucciones de movimiento por medio de un dispositivo a distancia, indicando la dirección hacia la que se tiene que dirigir. Tiene un ordenador de a bordo que controla su tracción, en función de las entradas que recibe de sus sensores.

Figura 20. Boston Dynamics Big Dog, controlado por el operador detrás.

iControlNao, es una aplicación para el sistema operativo de Apple, iOS, creada por Klaus Engel, un sistema de teleoperación supervisado que te permite, mediante funciones predeterminadas, realizar diversos movimientos en el robot NAO, como moverse, levantarse, etc. La aplicación detecta los robots Nao que están en la zona y se conecta automáticamente al que se seleccione, al parecer también se puede utilizar con el *software* de simulación de NAO. Se puede ver una imagen de la aplicación en la Figura 21.

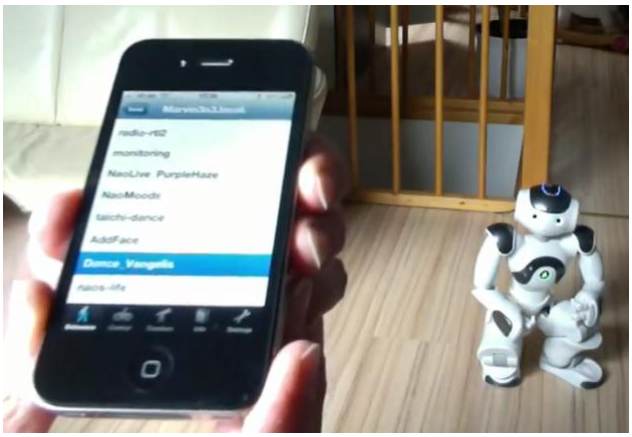

Figura 21. Aplicación iControlNao.

2.2.2 Interfaces

Una interfaz de control es un instrumento que permite a un operador, enviar instrucciones a un sistema para realizar una determinada tarea, puede definir las instrucciones necesarias para realizar las tareas para las que ha sido creado. A través de ella también es posible recibir retroalimentación proveniente del equipo/robot controlado, ya sean imágenes, sonidos o fuerzas. Estas pueden dividirse en 4 clases: directas, multisensoriales, de control supervisado y novel [9].

2.2.2.1 Directa

Es un tipo de interfaz en la cual el robot es controlado por medio de operadores manuales, como *joysticks*, mandos o aplicaciones de algún dispositivo electrónico o móvil. En algunos casos, el operador puede recibir información por medio de cámaras instaladas en el robot teleoperado, información visual del entorno en donde se encuentra. Con frecuencia son denominadas como interfaces de teleoperación "de dentro hacia afuera" (*inside-out driving piloting*) porque el operador se siente como si estuviera dentro del robot mirando hacia fuera. Recientemente este tipo de interfaces han sido utilizadas para reconocimiento de túneles en el alcantarillado, desactivación remota de minas y sistemas de tele-presencia submarina basados en video. Son las más apropiadas cuando es necesaria la toma de decisiones en tiempo real, y los sistemas de teleoperación soportan un retardo en las comunicaciones casi nulo. Aunque las interfaces directas pueden ser usadas fuera de estas condiciones, el resultado obtenido es sub-optimo. En particular, el control directo en presencia de retrasos es pesado y tiende a producir errores. Para reducir al mínimo el tiempo de aprendizaje, algunas interfaces directas ofrecen sistemas de control muy similares espacial y funcionalmente a vehículos que deben ser teleoperados. Muchos UAV's (vehículos aéreos no tripulados) son pilotados usando una cabina que simula el habitáculo de una aeronave. Otras interfaces intentan mejorar el rendimiento del operador por medio de un sistema de tele-presencia basado en video (utilizando un casco o gafas), y señales físicas. Al utilizar en este trabajo una interfaz directa, se va a hablar en mayor extensión de ejemplos de este tipo.

Una aplicación que hace uso de la teleoperación en Android, es la que se ve en la Figura 22, en donde se hace uso del acelerómetro de un móvil con android para mover las distintas partes del cuerpo del Nao. También se observa que se recibe la imagen de una cámara, pero no es la del robot, sino una que está situada en lo alto de la habitación que capta la escena.

Figura 22. Control de Nao por giroscopio.

Mahru, Figura 23, es un robot humanoide que puede ser teleoperado por medio de unos dispositivos con sensores, estos se colocan en las manos y la cabeza del usuario, y detectan la posición en la que está situado, transmitiéndola al robot, y este imita los movimientos del humano con un ligero retardo. También hacen uso de un sistema de cámaras situadas en la estancia en la que se encuentra el operador, capturando el movimiento del usuario, y poder transformarlo a movimientos en el robot. Es una interfaz directa donde el dispositivo maestro es el traje que lleva puesto el operador, y recibe imágenes por medio de unas gafas. Es un proyecto del Instituto de ciencia y tecnología de Korea (KIST) y Samsung Electronics.

Figura 23. Robot teleoperado Mahru.

En la Figura 24, se puede observar la tesis de licenciatura de Jonas Koenemann's donde, por medio de un traje con un sistema de Xsens MVN mocap (sistema de captura de movimientos) transforma los ángulos y la posición de las distintas articulaciones del operador a los ángulos en los que el robot se puede mover, casi en tiempo real. El robot activamente ajusta su centro de masa y sus articulaciones con cinemática inversa para no caerse.

Figura 24. Nao controlado por un traje Xsens MVN.

De los proyectos anteriormente citados, se ha obtenido la base para el desarrollo del sistema creado en este trabajo, en el que se hace uso de una aplicación en android para poder teleoperar el robot humanoide Nao. Todos estos proyectos se basan en un sistema de teleoperación mediante el concepto maestro/esclavo, donde el operador actúa como maestro y el robot como esclavo. Ya sea a través de sistemas con realimentación sensitiva para el usuario, como visión artificial, o utilizando otro tipo de dispositivos de control.

2.2.2.2 Multisensorial

Son aquellas interfaces que se utilizan cuando un robot opera en una situación compleja o altamente dinámica. Proveen al operador de una gran variedad de modos de control (actuadores individuales, movimiento coordinado, etc) y modos de información (de texto, visual, etc). Son muy útiles para sistemas que requieren acciones específicas en el contexto en el que se desenvuelven. Puede resultar difícil para el operador percibir correctamente el entorno remoto o tomar decisiones de control a tiempo. Por ello, se utilizan estas interfaces que resuelven estos problemas, en robots complejos con muchos sensores, en los que el usuario necesita mucha información y así operar correctamente sus distintas funcionalidades. Recientemente han sido utilizadas para la exploración de volcanes, servicios en satélites, y la operación de robots móviles con autonomía ajustable. Muestran información de diversas fuentes y numerosos sensores, presentándola en una vista única para ayudar al operador a entender el estado del robot. En vehículos teleoperados, pueden mejorar el conocimiento de la situación, facilitar un juicio más profundo, y aumentar la velocidad en la toma de decisiones.

Un ejemplo de este tipo de interfaz es RobUAlab [10] (Figura 25), sistema de teleoperación que forma parte del proyecto AutomatL@bs promovido por varias universidades españolas. Utiliza el software Easy Java Simulations y también está basado en Java y Java3D. Los comandos se pueden enviar a un robot real, situado en el laboratorio de la Universidad de Alicante para ver los resultados a través de Internet.

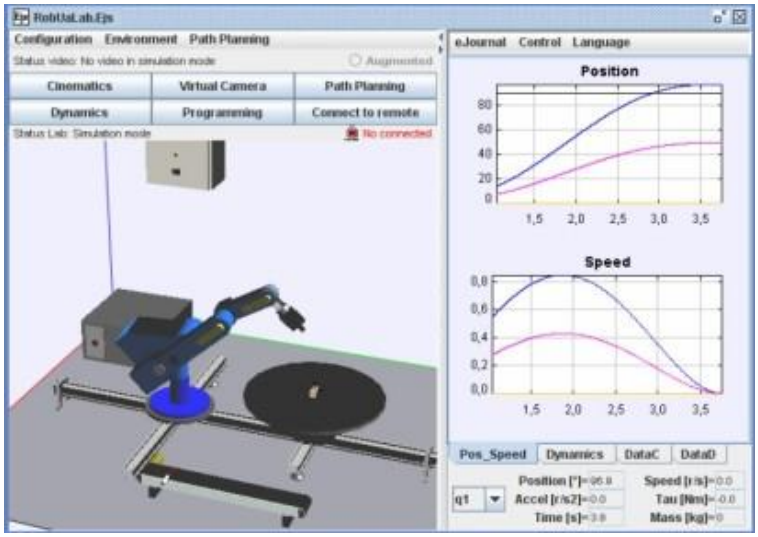

Figura 25. Interfaz multisensorial del simulador RobUAlab.

2.2.2.3 Control supervisado

Estas interfaces son diseñadas para la generación de comandos de alto nivel, monitorización y diagnóstico. Son muy adecuadas para aplicaciones que tienen grandes retrasos en las comunicaciones. Es común proporcionar facilidades para planificación de tareas y generación de secuencias (con frecuencia apoyadas por simulación y visualización en tiempo real). Además, a menudo proveen métodos para la revisión de los resultados, así el operador puede monitorizar e identificar anomalías en la ejecución. Para efectuar el control supervisado, el operador divide el problema en un conjunto de subtareas, que el robot puede ejecutar por él mismo. Lo que significa que el robot debe tener algún nivel de autonomía: ser capaz de alcanzar algunos objetivos mientras que se mantiene a salvo. En teleoperación de vehículos, el trabajo del operador está centrado principalmente en la navegación y la generación de comandos de movimientos. Hay muchos desafíos en el desarrollo de estas interfaces, como el diseño de la visualización de la información, la interacción entre el humano y el robot, y la compartición/intercambio de información. Por ejemplo, deben de proveer mecanismos para que el operador y el robot intercambien información a distintos niveles de detalle o abstracción. Es muy importante cuando el robot tiene problemas realizando una tarea y el operador tiene la necesidad de saber lo que ha ocurrido. Como ejemplo, cabe destacar el rover Sojourner que fue lanzado en 1996 hacia Marte (Figura 26).

Figura 26. Rover Sojourner.

2.2.2.4 Novel

El término novel (insólito/nuevo) es relativo: muchas de las interfaces de hoy en día fueron llamadas "novel", pero ahora son comunes. Por lo que es probable que las interfaces descritas a continuación dejen de llamarse novel en un futuro. Algunas son insólitas porque utilizan métodos poco convencionales de entrada. Por ejemplo:

- En el artículo de Terrence Fong *"Vehicle Teleoperation Interfaces"* [11], se describe una interfaz que utiliza las ondas cerebrales y la monitorización del movimiento de los músculos de una persona, para la conducción de un vehículo.

- En la *Carnegie Mellow University* (Figura 27), un mono controla un brazo robótico con su cerebro.

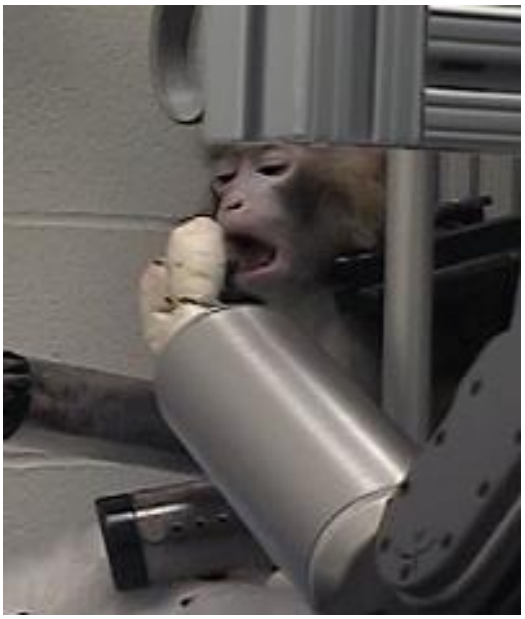

Figura 27. Universidad de Pittsburgh, mono controlando brazo robótico.

Las interfaces novedosas no solo se caracterizan por métodos poco convencionales de entrada, también son noveles si son usadas de forma inusual. En [12] describen un sistema que permite al operador "proyectar su presencia en un espacio real a distancia". En otras palabras, el vehículo teleoperado sirve como "avatar real" totalmente móvil para el operador.

Capítulo 3

Sistema de teleoperación

En este capítulo se realiza la descripción del sistema que ha sido desarrollado en este trabajo. Es un sistema de teleoperación con el que manejar el robot humanoide NAO, por medio de la utilización del sistema operativo ROS y el uso de dispositivos móviles con android.

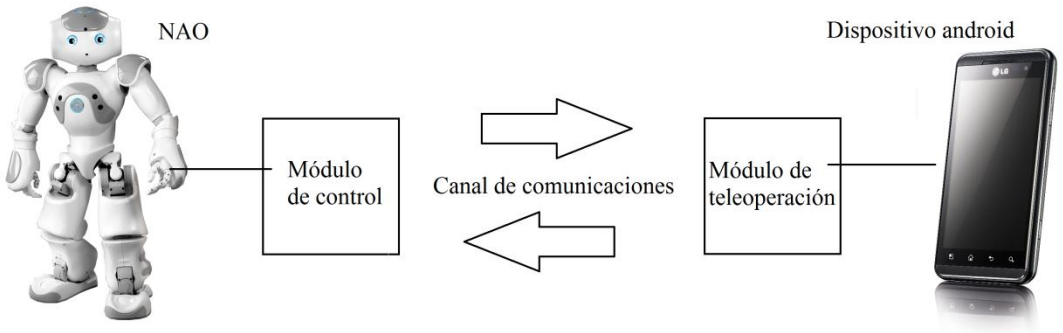

Figura 28. Esquema del sistema de teleoperación.

El sistema que ha sido desarrollado está formado por dos módulos, el primero de ellos es el de control, en el cual están implementados los procesos que gobiernan las acciones del robot, así como el sistema de comunicaciones al que se conecta el sistema de teleoperación, a través de un canal de comunicaciones inalámbrico. Este controlador es un híbrido de los sistemas de control de tipo supervisado y bilateral, puesto que por un

lado, el robot realiza una tarea de alto nivel, como es andar, no hay que decirle cómo mover las piernas, si no que directamente se le indica hacia donde tiene que avanzar, y el operador supervisa los movimientos que el robot realiza por medio de las imágenes que graba al mandarle las instrucciones a través del canal de comunicaciones. Y por otro lado existe cierto control bilateral al poder mover totalmente algunas partes del cuerpo como son los brazos completos y la cabeza, pero sin ningún tipo de realimentación de fuerzas hacia el operador.

El segundo módulo, denominado módulo de teleoperación, consiste en una aplicación interactiva que permite definir qué movimiento se tiene que realizar con el robot mediante un interfaz táctil. Además permite visualizar lo que ve el robot NAO, debido a que se reciben imágenes captadas por una de las cámaras del robot. Esta interfaz de control es de tipo directa, debido principalmente al tipo de dispositivos sobre los cuales va a ser ejecutada la aplicación. Es táctil, seleccionando las opciones de teleoperación de manera interactiva, utilizando unos controles, siendo la interacción con ellos muy visual. Debido a que los dispositivos móviles Android tienen una pantalla mediana o pequeña, sería casi imposible desarrollar una interfaz multisensorial en la que salgan todos los datos posibles del robot con gráficas y estadísticas. Además de no ser necesario puesto que lo único que se persigue con este trabajo es teleoperar el robot y recibir la información capturada mediante una de las cámaras del robot. Tampoco se podría implementar una interfaz de control supervisado ya que estos dispositivos no suelen tener visionado en 3D o ningún tipo de realimentación sensitiva hacia el usuario, exceptuando la vibración.

A continuación se presenta el proceso de análisis y diseño que se ha realizado para el desarrollo de la aplicación presentada en este documento.

3.1 Análisis

A continuación se presenta el diagrama de Casos de Uso, mediante este se describen cada una de las funcionalidades que ofrece el sistema de teleoperación. Posteriormente, los requisitos funcionales y no funcionales que han sido obtenidos de los diferentes casos de uso, así como de las restricciones del entorno operacional.

3.1.1 Diagrama de casos de uso

El sistema está constituido por 8 funcionalidades en total, entre las que realiza el usuario y las del robot. De estas 8, 3 son compartidas por ambos, es decir, mover piernas, cabeza, y brazos, las realizan los dos, puesto que en la interfaz el usuario va a mover unos *joysticks*, indicando la parte del cuerpo a mover por medio de unos botones, y el robot recibe las instrucciones de la interfaz y las ejecuta. En la Figura 29 se muestra el diagrama de casos de uso.

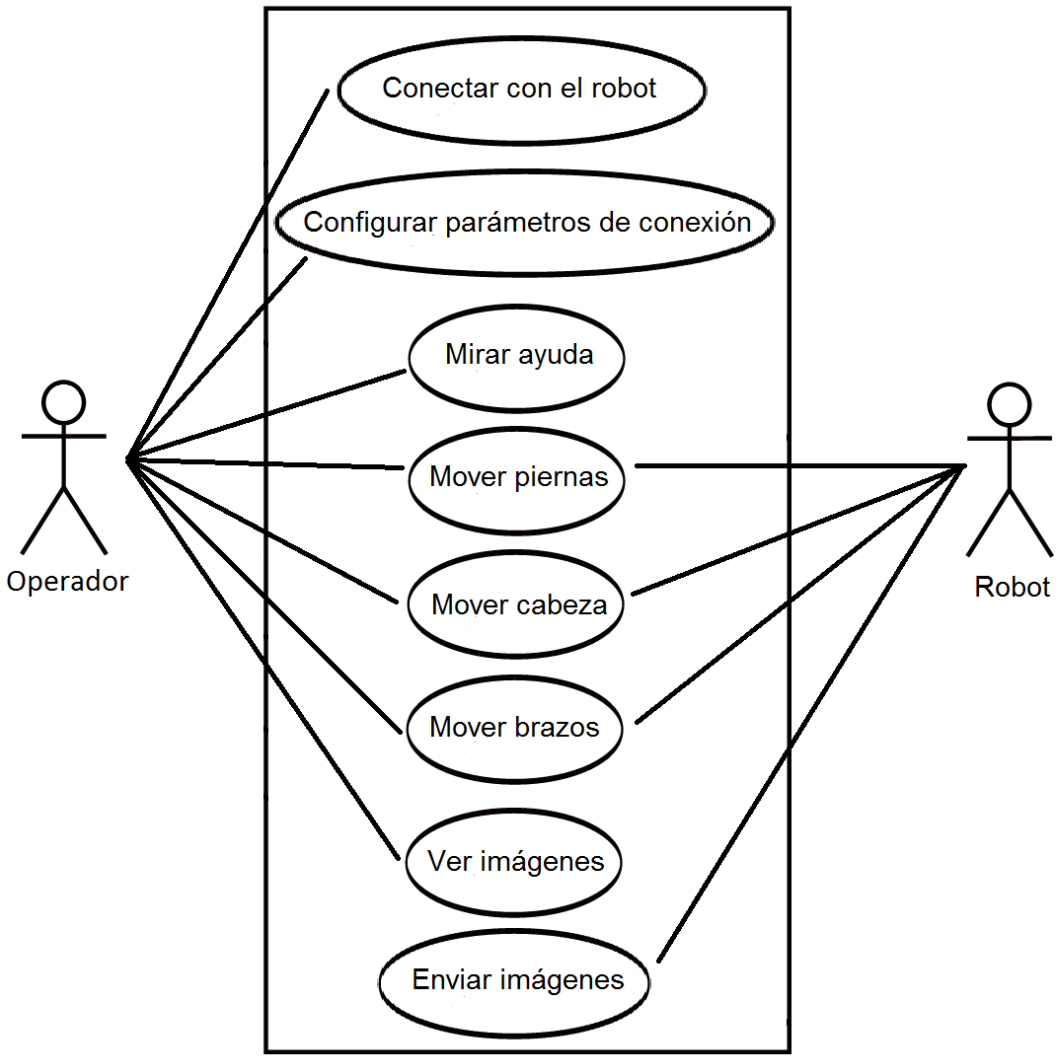

Figura 29. Diagrama de casos de uso.

3.1.1.1 Descripción de los atributos de los casos de uso

Para realizar las tablas en las que se describen textualmente los casos de uso, se han utilizado varios atributos representativos que se describen a continuación:

- Código: Identifica unívocamente un caso de uso, se construye añadiéndole a CU un "-" seguido de 3 dígitos. Ejemplo: CU-006.

- Nombre: Breve identificación del caso de uso.

- Actores: Conjunto de agentes que interactúan en el caso de uso. Los diferentes casos de usos son funcionalidades requeridas por los actores.

- Objetivo: Breve descripción de la finalidad del caso de uso.

- Precondiciones: Condiciones que deben cumplirse para poder realizar la funcionalidad del caso de uso.

- Postcondiciones: Estado en el que se queda el sistema tras realizar la funcionalidad requerida.

- Escenario básico: Descripción detallada de los pasos que sigue un actor al realizar una operación.

3.1.1.2 Descripción textual de Casos de Uso

En este apartado se realiza una descripción de los casos de uso representados en el diagrama de la Figura 29.

Código	CU-001
Nombre	Conectar con el robot.
Actores	Operador.
Objetivo	Conectar la aplicación *NaoController* al robot para poder posteriormente enviarle instrucciones de movimiento, o poder recibir imágenes por medio del nodo servidor de ROS.
Precondiciones	Tener los parámetros de conexión (IP y puerto) correctamente configurados y en funcionamiento el nodo *server*.
Postcondiciones	Conexión establecida entre *NaoController* y el nodo *server*.
Escenario básico	1. La aplicación *NaoController* muestra la interfaz principal en el dispositivo móvil. 2. El actor selecciona el botón del dispositivo móvil de configuración, para que le aparezca el sub-menú con la opción de “Conectar”. 3. La aplicación muestra el sub-menú con las diferentes opciones. 4. El actor selecciona “Conectar” para establecer la conexión entre la aplicación y el nodo de ROS. 5. La aplicación establece la conexión con el nodo servidor. 6. La aplicación comunica al actor por medio de un mensaje emergente de texto que la conexión se ha establecido.

Tabla 1. Descripción de Casos de Uso, Conectar con el robot.

Código	CU-002
Nombre	Configurar parámetros de conexión.
Actores	Operador.
Objetivo	Establecer los parámetros de conexión (IP y puerto) correctamente para poder conectarse posteriormente al nodo *server*.
Precondiciones	----
Postcondiciones	Parámetros de conexión correctamente establecidos.
Escenario básico	1. La aplicación *NaoController* muestra la interfaz principal en el dispositivo móvil. 2. El actor selecciona el botón del dispositivo móvil de configuración, para que le aparezca el sub-menú con la opción de "Ajustes". 3. La aplicación muestra el sub-menú con las diferentes opciones. 4. El actor selecciona "Ajustes" para modificar los parámetros de conexión. 5. La aplicación muestra una nueva pantalla en donde se encuentran los ajustes de conexión (IP, puerto y recepción de imágenes). 6. El actor introduce los datos de IP y puerto que tiene el nodo *server* de ROS por medio del teclado que le aparece al seleccionarlos. 7. La aplicación guarda los datos de conexión correctamente.

Tabla 2. Descripción de Casos de Uso, configurar parámetros de conexión.

Código	CU-003
Nombre	Mirar ayuda.
Actores	Operador.
Objetivo	Ver la ayuda de la aplicación para saber cómo realizar cada uno de los movimientos del cuerpo del robot por medio de la interfaz.
Precondiciones	----
Postcondiciones	El usuario ha visualizado correctamente la ayuda de la aplicación.
Escenario básico	1. La aplicación *NaoController* muestra la interfaz principal en el dispositivo móvil. 2. El actor selecciona el botón del dispositivo móvil de configuración, para que le aparezca el sub-menú con la opción de "Ayuda". 3. La aplicación muestra el sub-menú con las diferentes opciones. 4. El actor selecciona "Ayuda" para poder ver cómo se realizan los movimientos con la interfaz. 5. La aplicación muestra una nueva pantalla en donde se encuentran como se realizan cada uno de los movimientos de las distintas partes del cuerpo del robot. 6. El actor lee como se realizan los movimientos con la interfaz.

Tabla 3. Descripción de Casos de Uso, mirar ayuda.

Código	CU-004
Nombre	Mover piernas.
Actores	Operador.
Objetivo	Mover los *joysticks* de la interfaz para que la aplicación envíe un mensaje de movimiento de las piernas, y para que posteriormente el robot mueva las piernas y se desplace hacia la dirección seleccionada.
Precondiciones	Establecida la conexión entre la aplicación y el nodo de ROS.
Postcondiciones	Se manda un mensaje de movimiento de piernas al nodo *server* de ROS para que este lo publique en el robot.
Escenario básico	1. La aplicación *NaoController* muestra la interfaz principal en el dispositivo móvil. 2. El actor selecciona el botón de las piernas para poder moverlas. 3. La aplicación muestra el botón de las piernas como activado. 4. El actor selecciona y mueve el joystick izquierdo o el derecho para realizar un movimiento en las piernas del robot. 5. La aplicación envía un mensaje de movimiento de piernas al nodo de ROS para que este lo publique en el robot.

Tabla 4. Descripción de Casos de Uso, mover piernas humano.

Código	CU-005
Nombre	Mover piernas.
Actores	Robot.
Objetivo	Mover las piernas en la dirección publicada por el nodo *server* de ROS.
Precondiciones	Mensaje de movimiento de piernas enviado al nodo *server* desde la aplicación *NaoController*. Conexión establecida entre el nodo servidor y el robot por medio de los nodos de ROS correspondiente para poder publicar mensajes.
Postcondiciones	Desplazamiento del robot Nao en la dirección publicada por el nodo servidor.
Escenario básico	1. El nodo servidor recibe el mensaje de movimiento de piernas enviado por la aplicación. 2. Publica los datos espaciales que le han llegado en el mensaje para que puedan ser obtenidos por el robot. 3. El robot obtiene los datos de movimiento de las piernas en la dirección indicada por los mismos. 4. El robot actúa en consecuencia y se mueve en la dirección indicada.

Tabla 5. Descripción de Casos de Uso, mover piernas robot.

Código	CU-006
Nombre	Mover cabeza.
Actores	Operador.
Objetivo	Mover el joystick derecho de la interfaz para que la aplicación envíe un mensaje de movimiento de cabeza, y para que posteriormente el robot mueva la cabeza y esta se desplace hacia la dirección seleccionada.
Precondiciones	Establecida la conexión entre la aplicación y el nodo de ROS.
Postcondiciones	Se manda un mensaje de movimiento de la cabeza al nodo *server* de ROS para que este lo publique en el robot.
Escenario básico	1. La aplicación *NaoController* muestra la interfaz principal en el dispositivo móvil. 2. El actor selecciona el botón de la cabeza para poder moverla. 3. La aplicación muestra el botón de la cabeza, como activado. 4. El actor selecciona y mueve el joystick derecho para realizar un movimiento en la cabeza del robot. 5. La aplicación envía un mensaje de movimiento de cabeza al nodo de ROS para que este lo publique en el robot.

Tabla 6. Descripción de Casos de Uso, mover cabeza humano.

Código	CU-007
Nombre	Mover cabeza.
Actores	Robot.
Objetivo	Mover la cabeza en la dirección publicada por el nodo *server* de ROS.
Precondiciones	Mensaje de movimiento de cabeza enviado al nodo *server* desde la aplicación *NaoController*. Conexión establecida entre el nodo servidor y el robot por medio de los nodos de ROS correspondiente para poder publicar mensajes.
Postcondiciones	Movimiento de la cabeza del robot Nao en la dirección publicada por el nodo servidor.
Escenario básico	1. El nodo servidor recibe el mensaje de movimiento de cabeza enviado por la aplicación. 2. Publica los datos espaciales que le han llegado en el mensaje para que puedan ser obtenidos por el robot. 3. El robot obtiene los datos de movimiento de la cabeza en la dirección indicada por los mismos. 4. El robot actúa en consecuencia y mueve la cabeza en la dirección indicada.

Tabla 7. Descripción de Casos de Uso, mover cabeza robot.

Código	CU-008
Nombre	Mover brazos.
Actores	Operador.
Objetivo	Mover los *joysticks* de la interfaz para que la aplicación envíe un mensaje de movimiento de una de las articulaciones de un brazo, y para que posteriormente el robot mueva esa articulación hacia la dirección seleccionada.
Precondiciones	Establecida la conexión entre la aplicación y el nodo de ROS.
Postcondiciones	Se manda un mensaje de movimiento de una de las articulaciones de uno de los brazos al nodo *server* de ROS para que este lo publique en el robot.
Escenario básico	1. La aplicación *NaoController* muestra la interfaz principal en el dispositivo móvil. 2. El actor selecciona el botón de uno de los brazos para poder moverlo. 3. La aplicación muestra el botón del brazo seleccionado como activado. 4. El actor selecciona y mueve el joystick izquierdo para seleccionar la articulación. 5. El actor mueve el joystick derecho para realizar un movimiento en la articulación del robot seleccionada. 6. La aplicación envía un mensaje de movimiento de la articulación seleccionada al nodo de ROS para que este lo publique en el robot.

Tabla 8. Descripción de Casos de Uso, mover brazos humano.

Código	CU-009
Nombre	Mover brazos.
Actores	Robot.
Objetivo	Mover una articulación de un brazo en la dirección publicada por el nodo *server* de ROS.
Precondiciones	Mensaje de movimiento de brazo enviado al nodo *server* desde la aplicación *NaoController*. Conexión establecida entre el nodo servidor y el robot por medio de los nodos de ROS correspondiente para poder publicar mensajes.
Postcondiciones	Movimiento de una articulación de un brazo del robot Nao en la dirección publicada por el nodo servidor.
Escenario básico	1. El nodo servidor recibe el mensaje de movimiento de la articulación enviado por la aplicación. 2. Publica los datos espaciales que le han llegado en el mensaje para que puedan ser obtenidos por el robot. 3. El robot obtiene los datos de movimiento de la articulación en la dirección indicada por los mismos. 4. El robot actúa en consecuencia y mueve la articulación del brazo en la dirección indicada.

Tabla 9. Descripción de Casos de Uso, mover brazos robot.

Código	CU-010
Nombre	Ver imágenes.
Actores	Operador.
Objetivo	Activar la *checkbox* de recepción de imágenes para poder recibir las imágenes del robot Nao.
Precondiciones	Establecida la conexión entre la aplicación y el nodo de ROS. Conexión establecida entre el nodo servidor y el robot por medio de los nodos de ROS correspondiente para poder suscribirse y recibir el nodo servidor la imagen del robot.
Postcondiciones	Activada la casilla de recepción de imágenes para recibir las imágenes.
Escenario básico	1. La aplicación *NaoController* muestra la interfaz principal en el dispositivo móvil. 2. El actor selecciona el botón del dispositivo móvil de configuración, para que le aparezca el sub-menú con la opción de "Ajustes". 3. La aplicación muestra el sub-menú con las diferentes opciones. 4. El actor selecciona "Ajustes" para modificar la casilla de recepción de imágenes. 5. La aplicación muestra una nueva pantalla en donde se encuentran los ajustes de conexión y la casilla de recepción de imágenes. 6. El actor activa la casilla de recepción de imágenes. 7. La aplicación envía un mensaje de petición de imagen 8. La aplicación recibe la imagen por medio de un mensaje enviado por el nodo servidor de ROS

Tabla 10. Descripción de Casos de Uso, ver imágenes.

Código	CU-011
Nombre	Enviar imágenes.
Actores	Robot.
Objetivo	Enviar un mensaje con la imagen capturada por el robot Nao, obtenida por medio de la suscripción al nodo de ROS correspondiente, desde el nodo servidor.
Precondiciones	Mensaje de petición de imagen enviado al nodo *server* desde la aplicación *NaoController*. Conexión establecida entre el nodo servidor y el robot por medio de los nodos de ROS correspondiente para poder suscribirse al robot.
Postcondiciones	Envío de la imagen obtenida por el nodo servidor a la aplicación *NaoController*.
Escenario básico	1. El nodo servidor recibe el mensaje de petición de imagen enviado por la aplicación. 2. Se suscribe al nodo correspondiente de ROS que le permite recibir la imagen capturada por el robot. 3. El robot captura la imagen solicitada por una de sus cámaras. 4. El nodo servidor obtiene la imagen. 5. La envía por medio de un mensaje a la aplicación *NaoController*.

Tabla 11. Descripción de Casos de Uso, enviar imágenes.

3.1.2 Requisitos del sistema

En esta sección se describen los requisitos del sistema de teleoperación presentado en este documento. Estos se encuentran divididos en requisitos funcionales y no funcionales.

3.1.2.1 Descripción de los atributos de los requisitos

A continuación se presenta una explicación detallada de los campos elegidos para representar los requisitos funcionales y no funcionales:

- Código: Identifica unívocamente un requisito, se construye añadiéndole a una R, una F si es funcional, o NF si es no funcional, además de un "-" seguido de 3 dígitos. Ejemplo: RF-006 RNF-009.

- Descripción: Breve descripción del requisito.

- Módulo de aplicación: Indica sobre qué módulo se aplica el requisito.

- Necesidad: Determina si el requisito tiene que ser implementado obligatoriamente o de manera opcional. Los valores que puede tomar este campo son:

    - Esencial: El requisito tiene que ser implementado.

    - Deseable: Es preferible implementar el requisito, pero no es obligatorio.

    - Opcional: El requisito se podrá implementar, pero no es importante ni obligatorio.

- Prioridad: Medida de la importancia de los requisitos, para poder planificarlos correctamente en el proceso de diseño e implementación. Los valores que puede tomar este campo son:

    - Alta: El requisito debe ser implementado en el sistema al principio de su desarrollo.

    - Media: Debe ser añadido una vez acabados los requisitos de prioridad alta.

    - Baja: El requisito debe ser implementado después de añadir los de prioridad media.

- Estabilidad: Los requisitos pueden ser estables durante la vida útil del software. Sin embargo otros, pueden depender de cambios que se lleven a cabo en el diseño o la codificación. Los valores que puede tomar este campo son:

- Estable: El requisito no varía durante la vida del sistema.
- Inestable: El requisito varía a lo largo de la vida del sistema.

- Verificabilidad: Indica en qué grado es posible comprobar que el requisito se ha incorporado en el sistema creado. Los valores posibles en este campo son:
  - Alta: Se puede comprobar que el requisito se ha implementado en el sistema rápidamente.
  - Media: La incorporación en el sistema de este requisito se comprueba un poco peor que con una verificabilidad alta, pero también es fehaciente.
  - Baja: Es más difícil comprobar que el requisito se ha incorporado al sistema.

3.1.2.2 Requisitos funcionales

En este apartado se presentan los requisitos funcionales del sistema de teleoperación desarrollado.

Código	RF-001		
Descripción	Para usar la aplicación habrá un icono con el que se permita acceder pulsándolo.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 12. RF-001.

Código	RF-002		
Descripción	La aplicación debe mostrar una interfaz principal con la que el usuario maneje al robot.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 13. RF-002.

Código	RF-003		
Descripción	Se debe mostrar un menú de configuración al pulsar el botón de ajustes del dispositivo, en el que se encuentren los distintos botones de las opciones de la aplicación.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 14. RF-003.

Código	RF-004		
Descripción	Se debe mostrar un botón de conexión en el menú de configuración.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 15. RF-004.

Código	RF-005		
Descripción	Se debe mostrar un botón de ajustes de conexión en el menú de configuración.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 16. RF-005.

Código	RF-006		
Descripción	Se debe mostrar un botón de ayuda en el menú de configuración.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 17. RF-006.

Código	RF-007		
Descripción	Se debe mostrar un botón para salir de la aplicación en el menú de configuración.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 18. RF-007.

Código	RF-008		
Descripción	Se mostrará un mensaje de conexión establecida al pulsar el botón de conexión si la conexión es satisfactoria.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 19. RF-008.

Código	RF-009		
Descripción	Se mostrará un mensaje de conexión fallida al pulsar el botón de conexión si la conexión no se realiza correctamente.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 20. RF-009.

Código	RF-010		
Descripción	Se mostrará un mensaje de conexión cancelada al pulsar el botón de conexión si la conexión ya estaba establecida.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 21. RF-010.

Código	RF-011		
Descripción	Aparecerá un menú de ajustes, en el que se tengan las distintas opciones de conexión del robot, al pulsar el botón de ajustes en el menú de configuración.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 22. RF-011.

Código	RF-012		
Descripción	En el menú de ajustes de conexión se podrá cambiar la ip de conexión.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 23. RF-012.

Código	RF-013		
Descripción	En el menú de ajustes de conexión se podrá cambiar el puerto de conexión.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 24. RF-013.

Código	RF-014		
Descripción	Para introducir los valores de ip y puerto en el menú de ajustes de la conexión, se utilizará un teclado que saldrá en pantalla.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 25. RF-014.

Código	RF-015		
Descripción	En el menú de ajustes de conexión se podrá seleccionar si se quieren recibir las imágenes capturadas por el robot por medio de una caja de confirmación (*checkbox*).		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 26. RF-015.

Código	RF-016		
Descripción	Cada opción de ajustes de conexión tendrá una breve descripción.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 27. RF-016.

Código	RF-017		
Descripción	Se mostrará una pantalla de ayuda que aparezca al pulsar el botón de ayuda.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 28. RF-017.

Código	RF-018		
Descripción	En la ayuda se describirá cómo utilizar la interfaz para teleoperar el robot.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 29. RF-018.

Código	RF-019		
Descripción	Se saldrá de la aplicación al pulsar el botón de salir en el menú de configuración.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 30. RF-019.

Código	RF-020		
Descripción	Se seleccionarán las piernas del robot con un botón en la interfaz principal.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 31. RF-020.

Código	RF-021		
Descripción	Se seleccionará la cabeza del robot con un botón en la interfaz principal.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 32. RF-021.

Código	RF-022		
Descripción	Se seleccionará el brazo izquierdo del robot con un botón en la interfaz principal.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 33. RF-022.

Código	RF-023		
Descripción	Se seleccionará el brazo derecho del robot con un botón en la interfaz principal.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 34. RF-023.

Código	RF-024		
Descripción	Solo podrá haber seleccionado un botón de una de las partes del cuerpo del robot.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 35. RF-024.

Código	RF-025		
Descripción	Habrá dos *joysticks* en la interfaz principal para realizar los movimientos de las partes del cuerpo del robot.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 36. RF-025.

Código	RF-026		
Descripción	El movimiento que se estuviera realizando con alguno de los *joysticks* se dejará de realizar en cuanto se dejen utilizar.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 37. RF-026.

Código	RF-027		
Descripción	Se verán las imágenes capturadas por el robot en la interfaz principal, si se ha seleccionado la recepción de imágenes en el menú de ajustes de conexión, y si la conexión se ha establecido correctamente.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 38. RF-027.

Código	RF-028		
Descripción	Se podrá utilizar la aplicación tanto si el dispositivo está en horizontal o en vertical, ajustándose la interfaz a la orientación.		
Módulo de aplicación	Teleoperación		
Necesidad	Deseable	**Prioridad**	Media
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 39. RF-028.

Código	RF-029		
Descripción	La aplicación debe ser capaz de enviar mensajes de movimiento al nodo servidor, de la parte del cuerpo seleccionada, al mover los *joysticks*. Si la conexión está correctamente establecida.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 40. RF-029.

Código	RF-030		
Descripción	Mientras que no haya una conexión por parte de la aplicación, esperará a que se establezca.	**Módulo de aplicación**	Control
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 41. RF-030.

Código	RF-031		
Descripción	Debe de mostrar un mensaje para comprobar si la conexión con la aplicación en Android se ha establecido correctamente o si ha surgido algún problema.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 42. RF-031.

Código	RF-032		
Descripción	Debe de enviar al robot los mensajes de movimiento que recibe de la aplicación.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Baja

Tabla 43. RF-032.

Código	RF-033		
Descripción	Debe de enviar a la aplicación, las imágenes capturadas por una de las cámaras del robot.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 44. RF-033.

Código	RF-034		
Descripción	Debe esperar una nueva conexión si la aplicación cierra la actual.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 45. RF-034.

3.1.2.3 Requisitos no funcionales

En este apartado se presentan los requisitos no funcionales del sistema desarrollado.

Código	RNF-001		
Descripción	La aplicación debe ser desarrollada en Android.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 46. RNF-001.

Código	RNF-002		
Descripción	Debe ser compatible con la versión 2.1 de Android y superiores.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 47. RNF-002.

Código	RNF-003		
Descripción	Se utilizará un sistema de comunicación por medio de sockets TCP para el intercambio de mensajes con el nodo servidor.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	
Estabilidad	Estable	**Verificabilidad**	

Tabla 48. RNF-003.

Código	RNF-004		
Descripción	Debe haber un estándar para el intercambio de mensajes entre la aplicación y el nodo servidor.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Baja

Tabla 49. RNF-004.

Código	RNF-005		
Descripción	El dispositivo utilizado deberá disponer de conexión wifi.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 50. RNF-005.

Código	RNF-006		
Descripción	El idioma utilizado en la aplicación será el español.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 51. RNF-006.

Código	RNF-007		
Descripción	La aplicación deberá ser multitarea, para por ejemplo, recibir imágenes y mover los *joysticks* a la vez.		
Módulo de aplicación	Teleoperación		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Media

Tabla 52. RNF-007.

Código	RNF-008		
Descripción	El nodo servidor debe ser desarrollado en C++.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Media

Tabla 53. RNF-008.

Código	RNF-009		
Descripción	Se utilizará un sistema de comunicación por medio de sockets TCP para el intercambio de mensajes con la aplicación.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Baja

Tabla 54. RNF-009.

Código	RNF-010		
Descripción	Debe haber un estándar para el intercambio de mensajes entre el nodo servidor y la aplicación.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Baja

Tabla 55. RNF-010.

Código	RNF-011		
Descripción	El ordenador utilizado deberá disponer de conexión wifi.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 56. RNF-011.

Código	RNF-012		
Descripción	Deberá de existir una red wifi a la que conectarse.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 57. RNF-012.

Código	RNF-013		
Descripción	El idioma utilizado para los mensajes de conexión y error deberá ser el español.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Baja

Tabla 58. RNF-013.

Código	RNF-014		
Descripción	Para publicar los mensajes de movimiento tendrá que hacer uso de un publisher.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Baja

Tabla 59. RNF-014.

Código	RNF-015		
Descripción	Al publicar los mensajes de movimiento, tendrá que hacer uso del nodo nao_driver.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Media

Tabla 60. RNF-015.

Código	RNF-016		
Descripción	Para subscribirse a las imágenes capturadas por el robot tendrá que hacer uso de un subscriber.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Media

Tabla 61. RNF-016.

Código	RNF-017		
Descripción	Al subscribirse a las imágenes capturadas por el robot, tendrá que hacer uso del nodo nao_vision.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Media

Tabla 62. RNF-017.

Código	RNF-018		
Descripción	El ordenador utilizado tendrá el sistema operativo Ubuntu.		
Módulo de aplicación	Control		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Estable	**Verificabilidad**	Alta

Tabla 63. RNF-018.

3.1.3 Restricciones del sistema

A continuación se presenta la descripción de las restricciones que tiene el sistema de teleoperación. Estas se dividen en dos categorías, aquellas que son impuestas por el *hardware* de los dispositivos utilizados y las que provienen del *software* utilizado para el desarrollo de los distintos módulos del sistema.

3.1.3.1 Restricciones hardware

Son aquellas producidas por los dispositivos *hardware* que han sido utilizados para la realización de este trabajo. Por lo que se presentan cada una de las características requeridas, no todas las existentes.

- El robot NAO consta de motores eléctricos para poder mover cada una de sus articulaciones.

- El robot NAO dispone de dos cámaras para capturar imágenes, pero no es posible utilizar ambas al mismo tiempo, debido a la arquitectura *hardware* para la conexión de las cámaras.

- La conexión con el robot NAO se realizará por medio de una conexión inalámbrica. El robot posee una tarjeta inalámbrica que le permite conectarse a una red LAN o WAN.

- Los diferentes módulos que forman el sistema de teleoperación se comunican entre sí mediante una red LAN, debido a la estructura del sistema, esta red debe ser accesible mediante una conexión inalámbrica.

- El nodo servidor de ROS que envía las instrucciones de movimiento y las peticiones de imágenes al robot, se comunicará con él por medio de Wifi. Por lo que el equipo en el que se encuentre, debe tener una tarjeta de red Wifi.

- El dispositivo móvil en el que se instale la aplicación debe disponer de tecnología Wifi para poder realizar la conexión con el nodo servidor de ROS.

- El dispositivo móvil en el que se instale la aplicación debe de tener una pantalla táctil.

3.1.3.2 Restricciones software

Producidas por el entorno en el que se va a desarrollar el sistema de teleoperación. Es decir, los sistemas operativos, lenguajes de programación y *frameworks* utilizados.

- El *framework* de desarrollo para crear el nodo de control es ROS (*Robot Operating System*).
- El lenguaje de programación utilizado para el desarrollo del nodo servidor de ROS es C++, impuesto en parte por el *framework* utilizado (ROS).
- El *middleware* NaoQi, para facilitar el acceso a los sensores y actuadores en el desarrollo del nodo servidor en ROS.
- Para la comunicación con NaoQi se ha utilizado el paquete de funcionalidad ofrecido por la universidad de Friburgo [13]. Este ofrece un conjunto de librerías que facilitan la comunicación con el robot NAO.
- ROS debe ser ejecutado sobre un sistema operativo Ubuntu. La mejor alternativa sería la utilización de Ubuntu 10.04, 10.10 o 11.04, pero en nuestro caso ha sido utilizada la versión 10.04 para evitar posibles problemas de compatibilidad.
- Se debe usar el kit de desarrollo de Java (JDK), que se utilizará para la creación de la aplicación en Android, puesto que Android está basado en Java.
- El dispositivo móvil utilizado debe tener el sistema operativo Android.

3.1.4 Entorno operacional

A continuación se describen los distintos dispositivos *hardware* necesarios para la realización de este trabajo.

- Entorno operacional *hardware*. Se refiere a los distintos dispositivos *hardware* que son necesarios para poder usar el sistema desarrollado.
  - Robot humanoide NAO. Descrito de forma detallada en el apartado 3.1.4.1.
  - Un ordenador con tarjeta de red Wifi, en el que se tendrá el nodo servidor de ROS, para poder realizar el paso de mensajes entre la aplicación android y el robot. Este nodo podría ser instalado dentro del propio robot.
  - Un dispositivo móvil con tarjeta de red Wifi, para poder establecer una comunicación con el nodo servidor, y pantalla táctil.

- Un router inalámbrico.

- Entorno operacional *software*. Se refiere a los sistemas operativos, *frameworks*, librerías y *software* adicional necesario para desarrollar el sistema de teleoperación.

  - Sistema operativo Ubuntu 10.04, 10.10 o 11.04.

  - Versión Diamondback de ROS. Descrito con más detalle en el apartado 3.1.4.2.

  - Entorno de programación Eclipse, los entornos integrados de desarrollo (IDE) para Java y C++.

  - *Middleware* de programación NaoQi 1.10.52 o superior.

  - Entorno de programación y simulación Choregraphe 1.10.52 o superior.

  - Paquete de funcionalidades de la Universidad de Friburgo nao_0.3.

  - Kit de desarrollo Java (JDK) versión 6 o superior.

  - Kit de desarrollo de *software* android-sdk-r18 o superior.

  - Android 2.1 o superior para el dispositivo móvil o el simulador que se utilice.

3.1.4.1 Robot humanoide NAO

Nao es un robot humanoide autónomo programable, desarrollado por Aldebaran Robotics, empresa francesa con sede en París. El desarrollo del robot se inició con el lanzamiento del Proyecto Nao en el 2004. El 15 de agosto de 2007, Nao reemplazó a Aibo, perro robot de Sony, para utilizarlo como robot del Mundial de Fútbol de robots (Robocup) como plataforma estándar de la liga (SPL), una competición internacional de robótica. Nao se utilizó en la RoboCup de 2008 y 2009, y el NaoV3R fue elegido como plataforma para el SPL en la RoboCup de 2010.

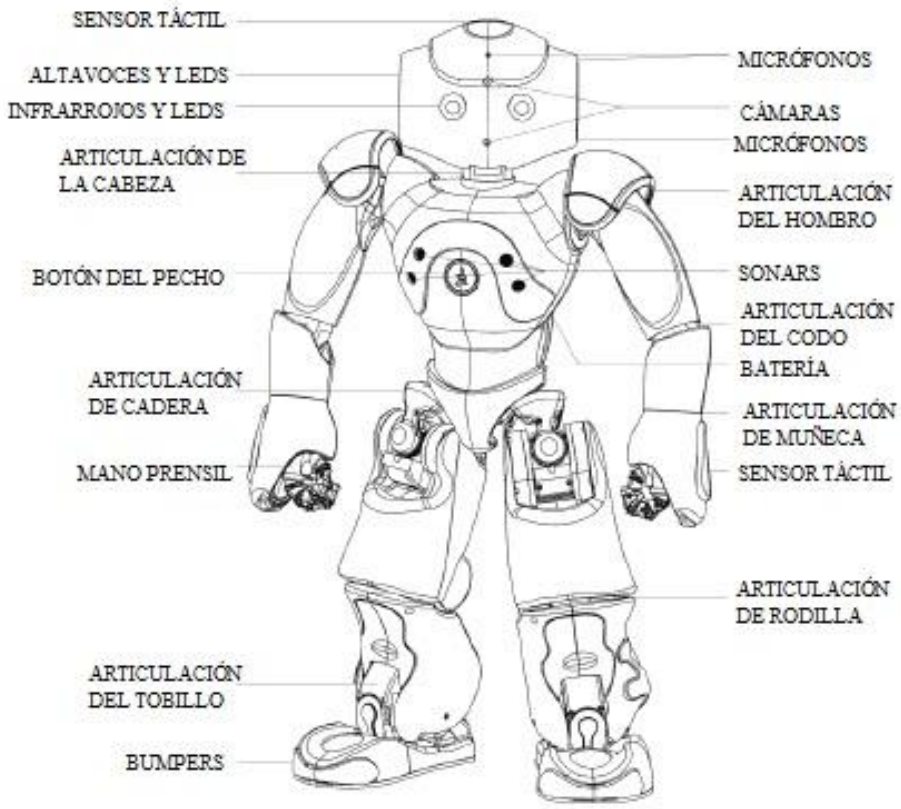

Figura 30. Características del robot humanoide NAO.

La edición de Nao Academic está disponible para universidades y laboratorios con fines educativos y de investigación. El robot NAO, presentado en la Figura 30, es un robot de tipo humanoide de 57 cm de altura y dispone de los siguientes sensores y actuadores [14]:

- Cuerpo con 25 grados de libertad, cuyos elementos clave son motores eléctricos y actuadores situados en las articulaciones.
- Un sensor inercial que le permite estabilizarse.
- Nueve sensores táctiles y ocho sensores de presión.
- Dos *bumpers* en los pies para detectar la colisión con los objetos.
- Un botón en el pecho de encendido/ apagado y para conocer el estado del robot.
- Dos cámaras VGA que graban 30 imágenes por segundo con las que puede reconocer objetos y caras.
- Cuatro micrófonos con los que puede reconocer el lenguaje y detectar la situación objetos en su entorno.

- Un sonar constituido por dos sensores emisores y receptores, por medio de los cuales detecta los objetos que se va encontrado.

- Varios dispositivos de comunicación, incluyendo un sintetizador de voz, luces LED, y dos altavoces de alta fidelidad con los que poder hablar.

- Una CPU Intel ATOM de 1,6 GHz, localizada en su cabeza, que ejecuta un *kernel* en Linux, y soporta *middleware* de Aldebaran (NaoQi).

- Una segunda CPU, localizada en el torso.

- Una batería de 27,6 vatios-hora de litio, que le proporciona una autonomía de 1,5 horas dependiendo del uso.

3.1.4.2 ROS

ROS (*Robot Operating System*) es un sistema operativo para robots, provee librerías y herramientas para ayudar a los desarrolladores de *software* a crear aplicaciones para robots. Fue originalmente desarrollado en el 2007 bajo el nombre de Switchyard por Morgan Quigley en el Laboratorio de Inteligencia Artificial de Standford, como soporte al proyecto robótico de IA de Standford. En 2008 el desarrollo continuó en el Willow Garage [15], un instituto de investigación robótica fuertemente comprometido con el desarrollo de código abierto y reutilizable, con más de 20 instituciones colaborando en modelos de desarrollo [16].

Su objetivo principal es permitir a los desarrolladores de *software* construir aplicaciones para robots de una manera más fácil y rápida para utilizarlas en plataformas comunes. Presta los servicios estándar de un sistema operativo, como abstracción de *hardware*, control de dispositivos de bajo nivel, implementación de la funcionalidad más frecuentemente utilizada, paso de mensajes entre procesos, y gestión de paquetes. Está basado en una arquitectura de grafos (Figura 31) en donde el procesamiento tiene lugar en los nodos que reciben, publican y multiplexan, el estado de los sensores, los controles, la planificación, los actuadores y otros mensajes. La librería está dirigida a sistemas Unix, aunque es soportada de manera experimental en Fedora y Mac OS X.

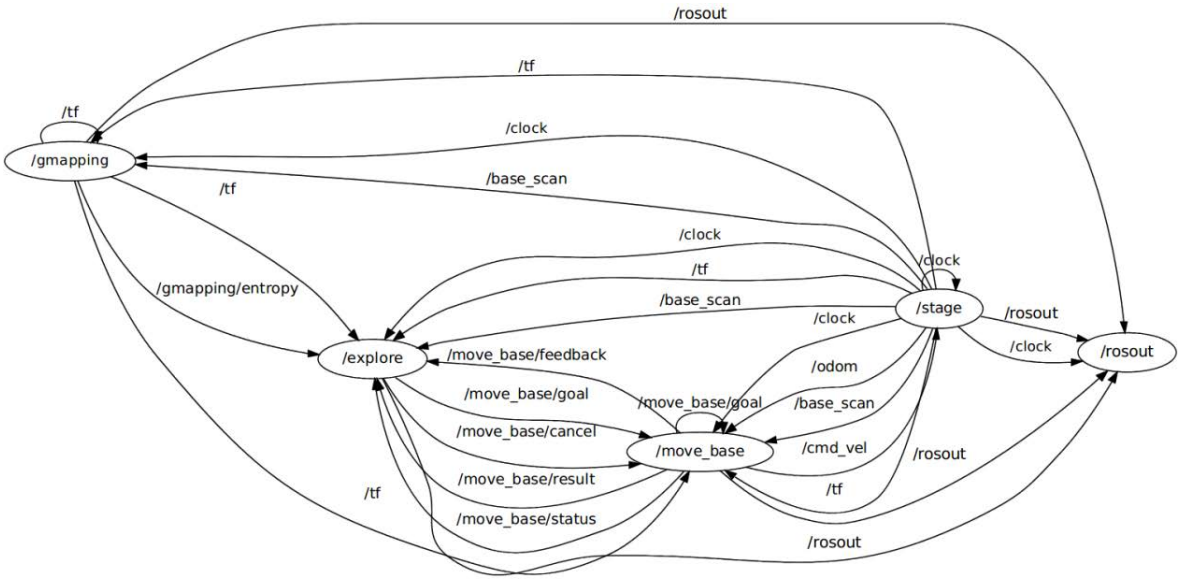

Figura 31. Ejemplo de la arquitectura de grafos que utiliza ROS.

ROS está formado por dos elementos básicos: El primero, el núcleo del sistema operativo y el segundo ros-pkg, un conjunto de paquetes realizados por usuarios que contribuyen con el desarrollo, organizados en grupos llamados *stacks* (pilas) que implementan funcionalidades como *mapping*, *planning*, percepción, simulación, etc.

ROS está realizado bajo una licencia BSD (*Berkeley Software Distribution*), y es *software* de código abierto. Es gratuito tanto para uso comercial como de investigación. Los paquetes de ros-pkg están bajo una gran variedad de licencias de código abierto.

Para desarrollar código en ROS, se pueden utilizar dos lenguajes de programación distintos, C++ y Python.

3.1.4.2.1 Componentes básicos

En ROS se pueden utilizar distintos tipos de comunicación, como la síncrona RPC mediante *Services*, comunicación asíncrona a través de los *Topics* y almacenamiento de datos por medio de *Parameter Server*. Los componentes básicos del sistema ROS son: Red ROS, Nodos (*Nodes*), Maestro (*Master*), Mensajes (*Messages*), Temas (*Topics*), Servicios (*Services*) [17].

- Red ROS: conjunto de nodos que forman un entramado de comunicación entre ellos, intercambiándose información mediante mensajes (Figura 32).

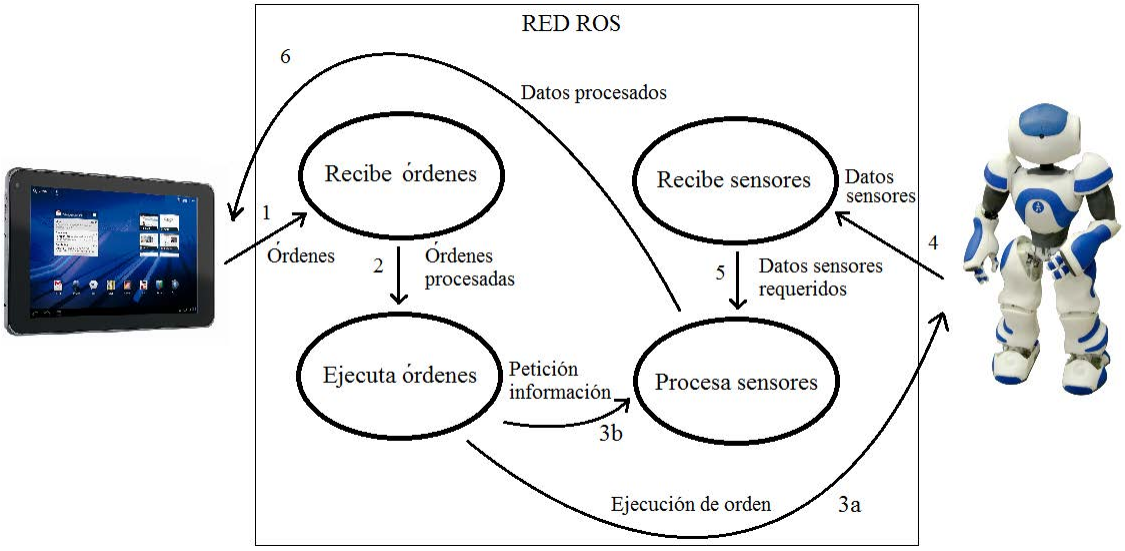

Figura 32. Ejemplo de red ROS.

- Nodos (*Nodes*): Son los elementos más pequeños que pueden encontrarse en una red ROS. Los nodos se comunican entre ellos utilizando los *topics* de transmisión, los servicios RPC (llamadas a procedimientos en máquinas remotas), y *Server Parameter* (Diccionario multi-variado compartido, utilizado para almacenar y recuperar parámetros en tiempo de ejecución del sistema). Los sistemas de control de robots suelen estar formados por muchos nodos, para abarcar todas las habilidades para las que el robot ha sido creado, por lo que se puede decir que ROS es un sistema modular. Como ejemplo se describe la Figura 32. Podría existir un sistema de teleoperación compuesto por varios nodos formando una red ROS. Uno llamado "Recibe órdenes", encargado de recibir las ordenes que se le van a enviar a un robot y procesarlas. Otro llamado "Ejecuta órdenes", que se comunique con el anterior y ejecute las órdenes enviándolas al robot, o con el nodo "Procesa sensores" para pedirle alguna información de los sensores. Un tercero llamado "Recibes sensores", encargado de recibir la información de los distintos sensores del robot. Y un cuarto llamado "Procesa sensores" que coja la información necesaria de los sensores y la procese para poder mostrarla al usuario.

- Maestro (Master): Es el servidor central de la arquitectura ROS, su función es permitir que todos los nodos se puedan localizar entre sí para comunicarse por medio de los *Topics* y *Services*, también es el encargado de suministrar los parámetros del servidor. En el ejemplo, permitiría que los nodos se localizaran y comunicaran.

- Mensajes (Messages): Es una estructura simple que contiene distintos tipos de datos. Los tipos básicos son enteros, decimales, booleanos, etc. También se pueden utilizar vectores y estructuras de datos. Son el medio de comunicación existente entre los nodos, se realiza mediante la publicación de mensajes que son transmitidos por medio de *Topics* o *Services*. En el ejemplo serían cada uno de los mensajes que se envían entre los nodos existentes.

- Canales (*Topics*): Son el medio por el que los nodos se intercambian mensajes, mediante un sistema de publicación y suscripción. Los nodos son de dos tipos:

    - Suscriptores (*Subscribers*): Se utiliza la suscripción a un determinado *topic*, en los nodos que están interesados en recoger información, para que posteriormente esta pueda ser tratada y realizar alguna operación con ella.

    - Publicadores (*Publishers*): Los nodos que generan información, la publican en el *topic* relevante correspondiente, para que otros nodos puedan posteriormente utilizarla.

  Puede haber varios suscriptores y/o publicadores de un mismo *topic*. La comunicación que se establece mediante los *topics* es unidireccional. Siguiendo con el ejemplo anterior, el nodo encargado de ejecutar las órdenes "Ejecuta órdenes", se suscribiría al *topic* que las publica, del nodo "Recibe órdenes" que las recibe. El nodo "Ejecuta órdenes" mandaría las órdenes correspondientes al robot, o publicaría las peticiones de información para que el nodo "Procesa sensores" se suscriba y las reciba. El nodo que recoge los datos de los distintos sensores "Recibe sensores", los publicaría en un *topic*, para que el nodo encargado de enviarlos al usuario "Procesa sensores" pudiera suscribirse a ellos.

- Servicios (*Services*): Los sistemas de solicitud/respuesta se realizan mediante un servicio, este se define por un par de mensajes: uno para la solicitud y uno para la respuesta. Un nodo proovedor de ROS ofrece un servicio bajo un nombre, y un cliente llama al servicio mediante el envío de un mensaje de solicitud y espera a la respuesta. Un cliente puede hacer una conexión persistente a un servicio, lo que permite un mayor rendimiento a costa de menos robustez en los cambios del proveedor de servicios.

3.1.4.2.2 Unidades de almacenamiento

El *framework* ROS agrupa el código que se implementa en *Packages* y *Stacks*, para hacer más fácil su posterior reutilización. Se describen a continuación ambos tipos de estructuras [17]:

- Paquetes (*Packages*): El *software* de ROS se organiza en paquetes. En un paquete puede haber distintos nodos de ROS, una biblioteca independiente de ROS, un conjunto de datos, archivos de configuración, *software* de terceros, o cualquier otra cosa que constituya un módulo útil. El objetivo de estos paquetes es proporcionar las funcionalidades de una manera fácil de utilizar, para que este *software* sea fácil de reutilizar. En general, los paquetes de ROS siguen el principio de "Goldilocks" (ricitos de oro): funcionalidad suficiente para ser útil, pero no demasiada para que el paquete no sea pesado y difícil de usar desde otro *software*.

- Pilas (*Stacks*): Son conjuntos de *packages* que juntos abarcan una funcionalidad completa. Por ejemplo, se podrían reunir en un *stack* todos aquellos *packages* que permiten que un robot mueva cada una de sus articulaciones, en donde cada *package* se encargaría de una parte distinta del robot.

3.2 Diseño

En este apartado se presenta el diseño arquitectónico de la aplicación.

- En el apartado 3.2.1 se realiza una introducción de manera general a la arquitectura del sistema de teleoperación.

- En el 3.2.2 se define como se ha diseñado el nodo de ROS encargado de enviar y recibir los mensajes del robot humanoide NAO.

- En la subsección 3.2.3 se describe el diseño de la interfaz de la aplicación en Android que se llamará *NaoController*, junto con cada uno de los elementos que las conforman.

3.2.1 Diseño general de la arquitectura

A continuación se va a describir de manera general el diseño que tiene la arquitectura global del sistema creado (Figura 33).

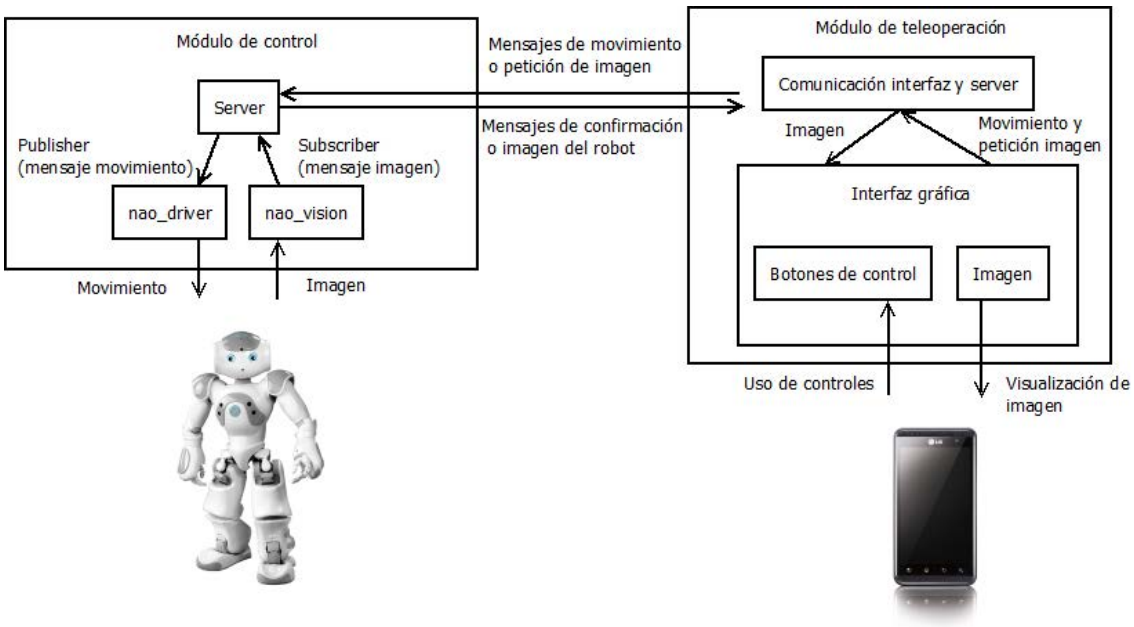

Figura 33. Diseño de arquitectura global.

Se pueden distinguir dos partes:

Módulo de control

El módulo de control es el encargado de gestionar los movimientos y las peticiones de imagen que le llegan al robot. Se encuentra dividido en tres módulos o sistemas:

- Nodo server: Por medio del cual se reciben los mensajes de movimiento y peticiones de video de la aplicación *NaoController*. Y se mandan los mensajes de confirmación de envío, para avisar a la aplicación que el servidor está listo para recibir mensajes, y los mensajes de imagen, en los cuales se encuentran las imágenes que el robot captura. Este interactúa con otros dos nodos en ROS descritos a continuación.

- Nodo nao_driver: Es el que permite al *server* enviar los mensajes de movimiento de cada una de las partes del cuerpo al robot, así como si fuese preciso, también podría dar la información de las mismas, aunque en este caso no sea necesario.

- Nodo nao_vision: Se encarga de enviar al nodo servidor las imágenes que el robot captura por medio de una de sus cámaras.

Módulo de teleoperación

El módulo de teleoperación es el medio de interacción entre el usuario y el robot, encargado, a su vez, de enviar los mensajes de movimiento y petición de imagen al módulo de control. Se encuentra dividido en dos módulos:

- Interfaz gráfica. Se pueden encontrar los ajustes para poder realizar la conexión o seleccionar la recepción de imágenes, la ayuda para ver cómo mover cada una de las partes del cuerpo del robot, la actividad principal, en donde se pueden ver las opciones de conexión y cada uno de los botones de selección de las partes del cuerpo del robot y de control del mismo, y la visualización de las imágenes recibidas.

- Comunicaciones. Se encarga de enviar los mensajes de petición de movimiento, movimiento y petición de imagen del robot al nodo servidor, provenientes de la interacción del usuario con la interfaz. Y recibir los mensajes de confirmación de movimiento y de imágenes, para poder visualizar estas últimas en la interfaz gráfica.

3.2.2 Modelo de comunicaciones

El módulo de control y el módulo de teleoperación se intercambian mensajes para que el robot humanoide Nao actúe de acuerdo a las indicaciones que el usuario de la aplicación ejecute por medio de la interfaz, y sea posible recibir las imágenes que el robot capta a través de una de sus cámaras. En la Figura 34 se observa la arquitectura de comunicaciones existente entre los distintos dispositivos que se utilizan en el sistema de teleoperación.

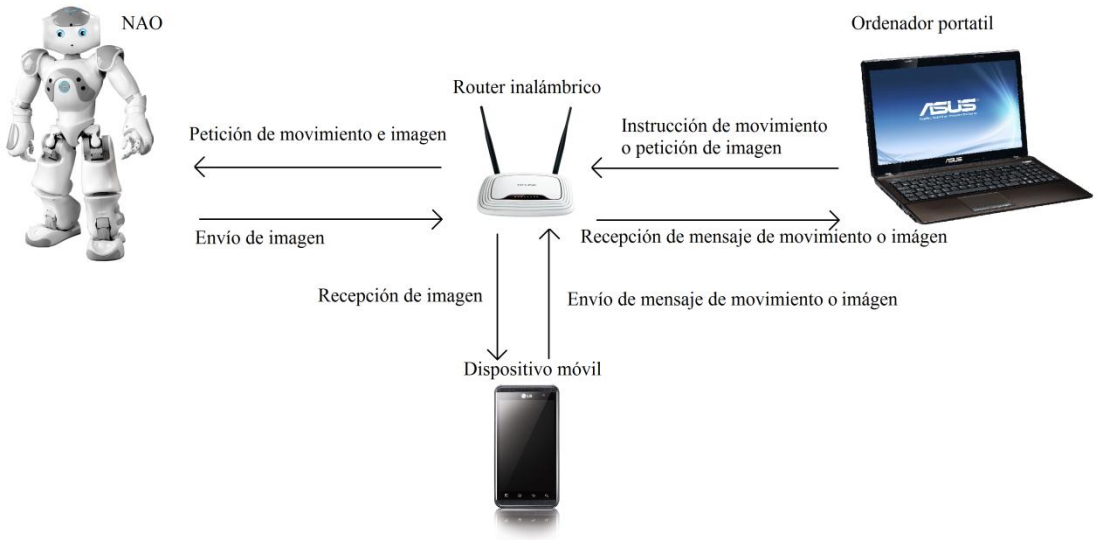

Figura 34. Arquitectura de comunicaciones.

Aclarar que este sistema está pensado para actuar cuando todos los dispositivos son accesibles entre sí a través de una misma red Wifi. Pero se podría utilizar a través de internet si se tuvieran dos ip's dedicadas (únicas), una para el dispositivo con Android y otra para el pc en el que se encuentra el servidor que envía las instrucciones al robot, siempre y cuando las direcciones ip's de los dispositivos sean conocidas. Además este sistema admite varias configuraciones, una con tres ip's cuando el módulo de control no está implantado en el robot y otra con dos ip's cuando el módulo está dentro del robot.

A continuación se describen los mensajes que se intercambian para producir la comunicación en orden:

1. Mensaje de confirmación: Después de producirse la conexión entre el módulo de teleoperación y el módulo de control, este último envía un mensaje de confirmación al de teleoperación, para que pueda enviar un mensaje de cualquier tipo (movimiento o petición de imagen). El mensaje de confirmación tiene la siguiente estructura: "ok".

2. Mensaje de operación: El módulo de teleoperación envía un mensaje con el tipo de instrucción que se va a enviar al módulo de control. Este define que parte del cuerpo del robot se mueve o si se quiere recibir la imagen capturada por el robot. En la Tabla 64 se muestran los distintos tipos de mensaje que pueden ser enviados. A continuación se describen los campos de esta:

    - Cabecera: Es la letra o conjunto de letras que se utilizan para distinguir unívocamente el tipo de mensaje que se está enviando.

    - Descripción: Explicación breve y concisa de cada una de las cabeceras que se pueden enviar.

Cabecera	Descripción
p	Desplazamiento con las piernas hacia cualquier dirección.
g	Giro sobre sí mismo con las piernas hacia cualquier dirección.
c	Movimiento de la cabeza.
ih	Movimiento del hombro izquierdo.
ic	Movimiento del codo izquierdo.
in	Movimiento de la muñeca izquierda.
im	Movimiento de la mano izquierda.
dh	Movimiento del hombro derecho.
dc	Movimiento del codo derecho.
dn	Movimiento de la muñeca derecha.
dm	Movimiento de la mano derecha.
pv	Petición de imagen capturada por el robot.

Tabla 64. Cabeceras de mensajes de movimiento.

1. Si el mensaje es de operación (Figura 35):

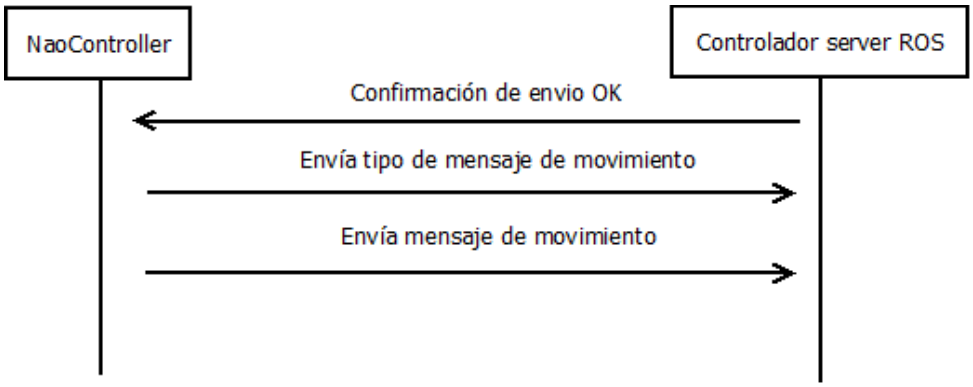

Figura 35. Paso de mensajes de movimiento.

2.1 Mensaje con valores: el servidor se queda esperando a recibir el mensaje con la instrucción de movimiento, el cliente la envía y el servidor la recibe y la publica en el robot. El mensaje consta de las siguiente partes:

- X: Es el valor correspondiente con el eje de abscisas producido por el movimiento de uno de los *joysticks* en la interfaz de la aplicación *NaoController*. Ej: "0.5".
- Y: Es el valor correspondiente con el eje de ordenadas producido por el movimiento de uno de los *joysticks* en la interfaz de la aplicación *NaoController*. Ej: "0.6".
- Separador: Sirve para separar cada uno de los valores mencionados anteriormente y poder reconocerlos al recibirlos. Se representan con el símbolo "|". Ej: "0.5|0.6" que corresponde con un mensaje de 0.5 en el eje X y 0.6 en el eje Y.

2. Si el mensaje es de petición de video (Figura 36):

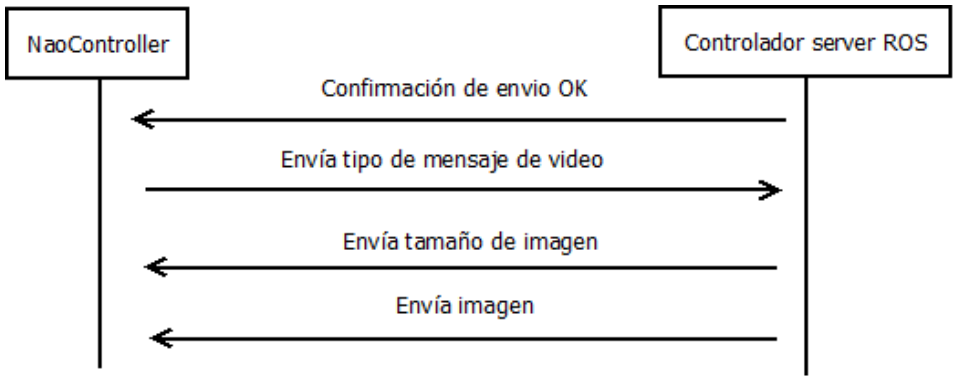

Figura 36. Paso de mensajes de video.

2.1 Tamaño de imagen: el servidor envía al cliente un mensaje con el tamaño de la imagen que ha capturado el robot, el cliente lo recibe, y se queda esperando a recibir la imagen con el tamaño especificado. Ej: "8567" es el tamaño total de la imagen que se va a enviar.

2.2 Mensaje de imagen: El servidor envía la imagen y el cliente la recibe y la muestra al usuario por la interfaz. Lo que se envía es un *array* del tamaño indicado anteriormente con el valor numérico del color correspondiente a los pixeles de la imagen. Ej: " 245 4 38 64 … 134 2 90".

3.2.3 Diseño del módulo de control

A continuación se realiza una descripción detallada del diseño del nodo servidor desarrollado en ROS, enumerando todos sus detalles. La Figura 37 muestra el conjunto de nodos que interactúan con el nodo *server* dentro del módulo de control, descritos en el apartado 3.2.1.

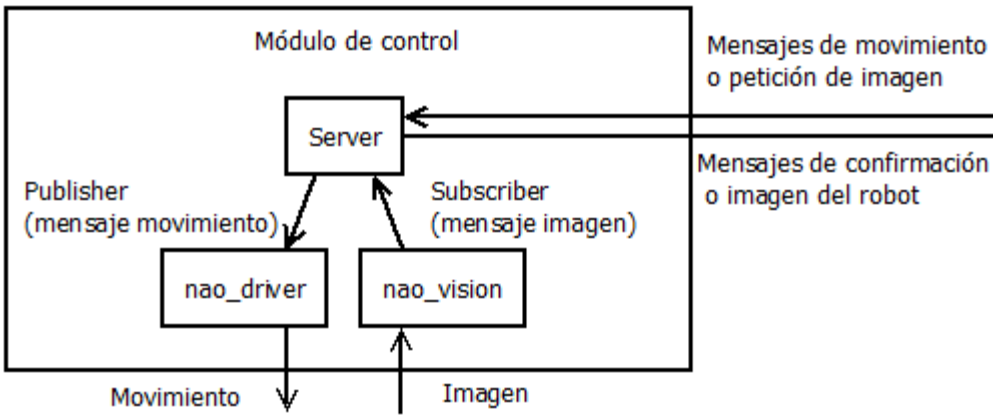

Figura 37. Módulo de control.

A continuación se explican las funcionalidades que ofrece el módulo de control:

1. Al iniciarse el nodo servidor, espera a que un usuario establezca una conexión por medio de la aplicación *NaoController*.

2. Una vez conectado, al nodo servidor le llegarán peticiones de movimiento y de envío de imágenes desde la aplicación Android. Este las tratará en función del tipo que sean.

    2.1 Si son de movimiento, publica el movimiento de la articulación que se quiera mover por medio de un nodo llamado nao_driver. Este proporciona acceso al movimiento de las piernas, los ángulos de las articulaciones y datos de los sensores del robot.

2.2 Si es petición de imagen se suscribe a las imágenes que graba el robot, por medio del nodo nao_vision, para enviarlas de vuelta a la aplicación. Este provee acceso a las imágenes que el robot captura por medio de una de sus dos cámaras.

3. Al finalizar el tratamiento del mensaje, vuelve a esperar otro.

4. Si la aplicación se desconecta, el servidor pasa a esperar nuevamente una conexión.

Cabe mencionar las siguientes restricciones del módulo de control:

1. La única manera que se tiene de saber el estado de la conexión, es por medio de los mensajes básicos que se imprimen por la terminal, al lanzar el nodo servidor, se muestran en la Figura 38. Informan del estado de la conexión con un cliente (aplicación). No existe una interfaz gráfica en el lado del servidor.

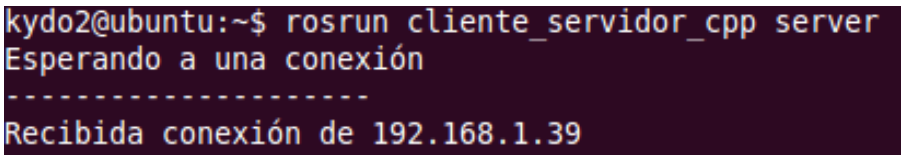

Figura 38. Mensajes de texto del nodo server.

2. Para poder realizar las funcionalidades requeridas para este trabajo, a partir de una primera aproximación de un servidor inicial, en el que solo se publicaba un tipo de movimiento enviado por un cliente simple en Java, se creó un estándar para identificar los mensajes, descrito anteriormente en el apartado 3.2.2, para poder diferenciar a que parte del cuerpo del robot iba destinado el mensaje de movimiento o cuándo era una petición de imagen enviada por parte de la aplicación *NaoController*.

3. Si un usuario cierra la aplicación o se desconecta, el servidor recibe un mensaje de desconexión, para dejar de esperar mensajes de ese cliente.

4. El servidor es multihilo, por lo que soporta múltiples conexiones, ya que se podrían conectar varios clientes y controlar distintas partes del mismo robot, el problema que conlleva es que si no se ponen de acuerdo en que partes controlar, podrían surgir problemas de coordinación. Esta posibilidad no ha sido estudiada en este trabajo actual y se propone como trabajo futuro.

3.2.3.1 Diagrama de funcionamiento

A continuación, en la Figura 39, se representa el diagrama de flujo del nodo *server*, en el que se muestra cómo funciona el flujo de datos en el servidor.

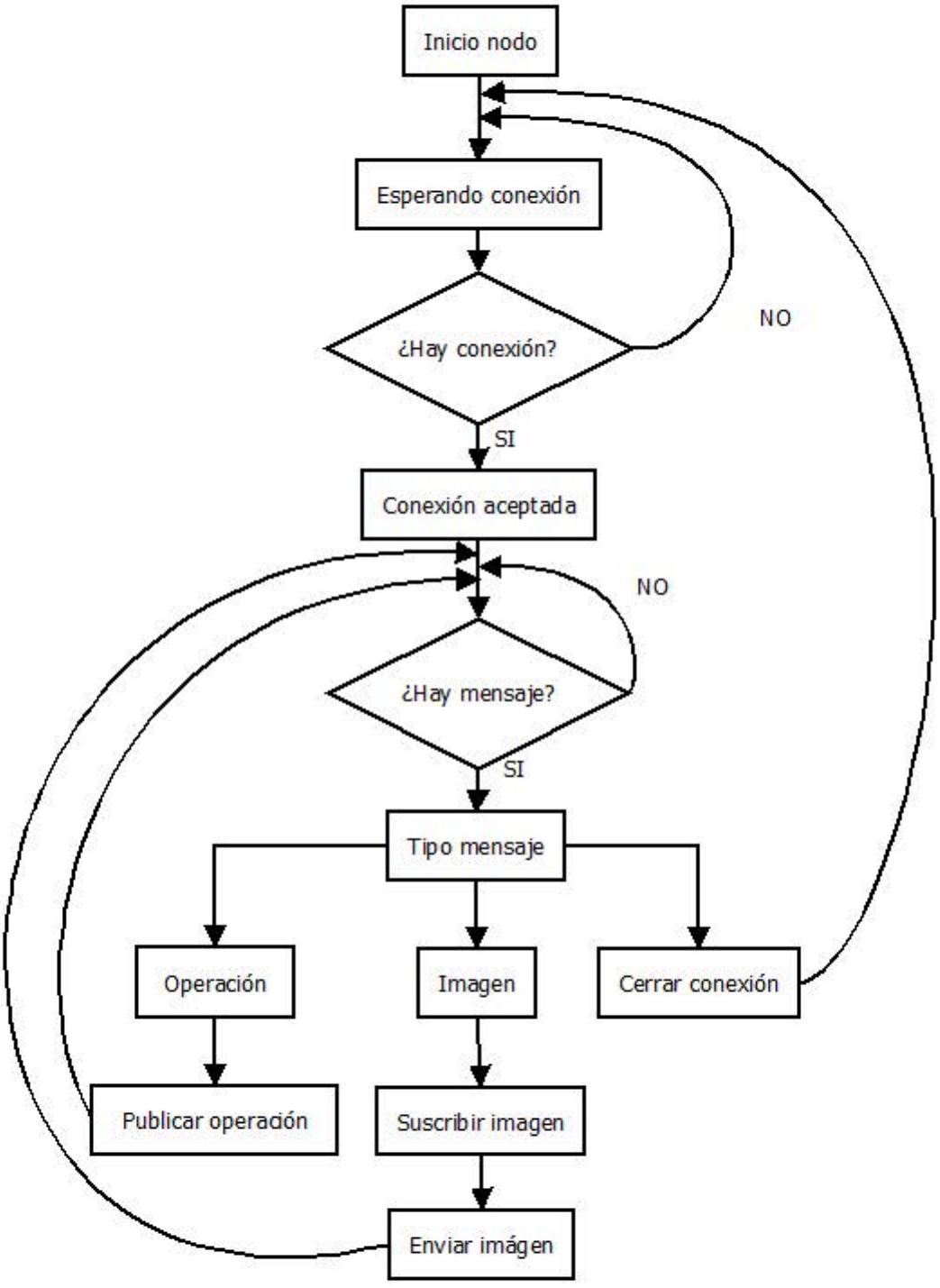

Figura 39. Diagrama de flujo del servidor.

Seguidamente se describe el flujo que sigue el nodo servidor. Desde que se inicia, y se conecta un cliente, hasta que se cierra la conexión y se queda esperando al siguiente en conectar.

1. El programa se inicia en la funcionalidad “Inicio nodo”.

2. Se queda esperando la conexión de un cliente en la funcionalidad “Esperando conexión”, preguntando si hay una conexión “¿Hay conexión?”.

3. Si hay una conexión y se acepta correctamente pasa a la funcionalidad “Conexión aceptada”, y espera a que le llegue un mensaje “¿Hay mensaje?”.

4. Llega un mensaje cuando pasa a la tarea “Tipo mensaje”.

5. Cuando el mensaje “Tipo mensaje” es de operación “Operación”.

    - Asigna los valores de los ejes X e Y que le han llegado a la articulación o parte del cuerpo correspondiente, y lo publica por medio del nodo nao_driver. Publica la operación “Publicar operación”, y vuelve a 3, esperando un nuevo mensaje “¿Hay mensaje?”.

6. Cuando el mensaje “Tipo mensaje” es de petición de imagen “Imagen”.

    - Se suscribe a la imagen que graba el robot “Suscribir imagen”, que proporciona el nodo nao_vision.

    - La envía a la aplicación “Enviar imagen” y vuelve a esperar a que llegue un mensaje “¿Hay mensaje?” en 3.

7. Si el mensaje “Tipo mensaje” es para avisar que el cliente va a desconectarse en la funcionalidad “Cerrar conexión”, deja de esperar mensajes y vuelve a esperar una conexión de otro cliente en la funcionalidad 2, “Esperando conexión”.

3.2.4 Diseño del módulo de teleoperación

A continuación se presenta el diseño final del módulo de teleoperación en Android, denominado *NaoController*. En la Figura 40 se presenta el módulo de teleoperación con cada uno de sus componentes, descritos en el apartado 3.2.1.

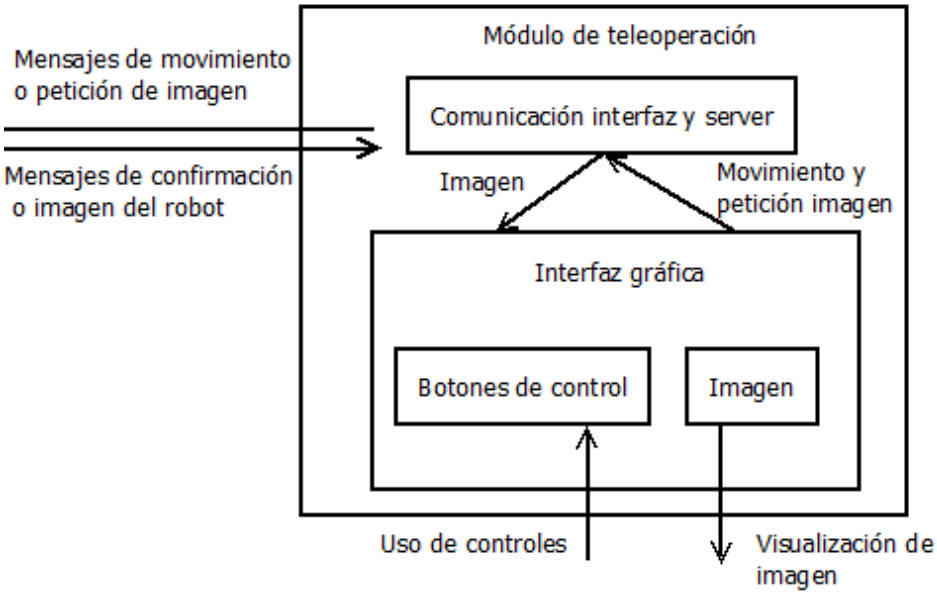

Figura 40. Módulo de teleoperación.

La aplicación se ha creado para ser usada en versiones de Android iguales o superiores a la 2.1, para aumentar la compatibilidad con modelos de móviles más antiguos, ya que la última versión disponible del sistema operativo Android es la 4.1. El sistema de mensajes es el explicado anteriormente en el apartado 3.2.2. A continuación se describen las características de la interfaz gráfica para controlar el robot:

3.2.4.1 Interfaz principal

A continuación se muestra la interfaz desarrollada (Figura 41).

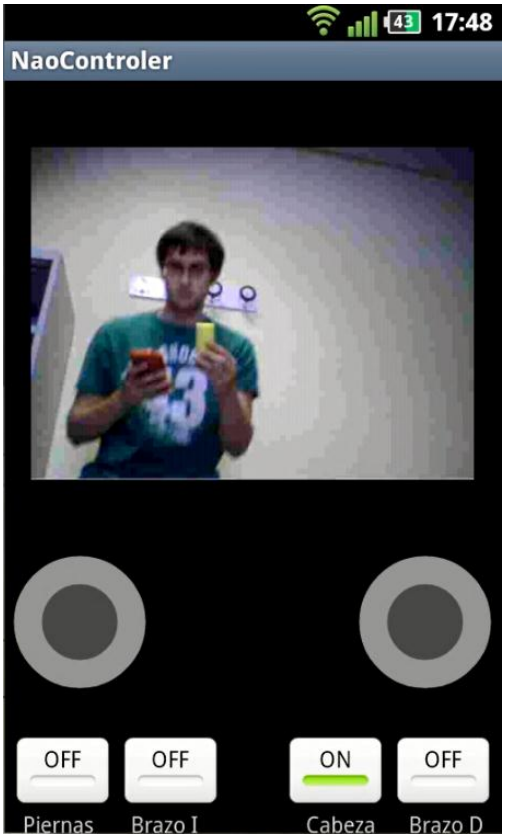

Figura 41. Interfaz final.

- Se tienen cuatro botones en la parte inferior de la interfaz, que se utilizan para seleccionar qué parte del cuerpo del robot se quiere mover. De izquierda a derecha, las piernas, el brazo izquierdo, la cabeza o el brazo derecho. Debido a la complejidad de los movimientos a realizar en la interfaz, solo se puede activar una de las partes del cuerpo del robot cada vez que se quiera realizar un movimiento, puesto que en todos los movimientos menos el de cabeza, es necesario el uso de los dos *joysticks* para mover una única parte del cuerpo. Para que no hubiera confusiones en la interfaz, cuando se selecciona una parte del cuerpo, si hubiera alguna seleccionada anteriormente, esta pasa a estar desactivada.

- Dos *joysticks* a ambos lados de la pantalla, de forma que puedan ser accionados mediante los pulgares, facilitando su uso.

    - Para mover las piernas, con el botón de las piernas activado, se usa el joystick izquierdo para desplazar al robot en cualquier dirección y el derecho para rotarlo sobre sí mismo.

    - Para mover las articulaciones de los brazos, con el botón de uno de los brazos activado, se seleccionan de la siguiente manera: Moviendo el joystick izquierdo hacia arriba se selecciona el

hombro, hacia abajo el codo, hacia la izquierda la muñeca y hacia la derecha la mano. Y con el derecho se mueve la articulación seleccionada.

- Para mover la cabeza, con el botón de la cabeza activado, se utiliza el joystick derecho desplazándolo en la dirección deseada.

- Se recibe como imagen, la grabación que el robot hace por medio de una de sus cámaras, como se puede observar en la parte superior de la interfaz.

3.2.4.2 Menú de configuración

A continuación se muestra el menú desarrollado (Figura 42).

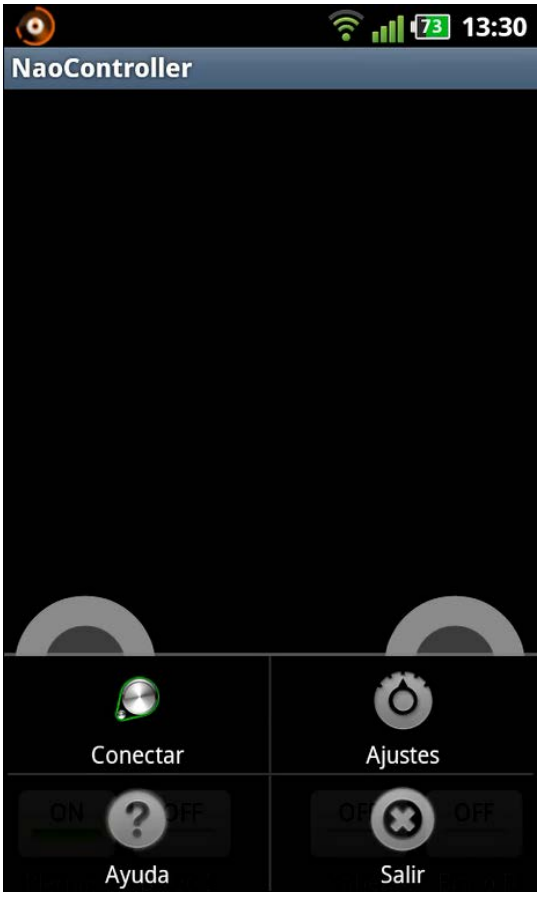

Figura 42. Ajustes de interfaz final.

- Se establece o cancela la conexión con el servidor pulsando el botón "Conectar".

- Se ajustan los parámetros de conexión pulsando el botón de "Ajustes". Despliega una serie de opciones para configurar correctamente los datos de conexión con el nodo servidor. Se pueden observar en la Figura 43. Dentro de las preferencias de conexión se pueden encontrar 3 campos, IP, puerto y Cámara de Nao.

Figura 43. Diseño de ajustes de la aplicación.

- En el campo IP se introduce la dirección ip del nodo servidor de ROS al que se va a conectar la aplicación para enviar y recibir mensajes, tanto de movimiento, como de video. Al pulsar el campo "IP", aparece un nuevo recuadro donde escribir la dirección ip del servidor y un teclado táctil con el que poder introducirla, como se observa en la Figura 44. Al iniciar la aplicación aparece una por defecto, pero al introducir nuevos valores, el último es el que se queda guardado en el campo IP, para que sea más fácil conectar en posteriores ocasiones.

Figura 44. Introducir ip en preferencias.

- En el campo puerto, se introduce el puerto en el que está escuchando el nodo servidor, para poder enviarle los mensajes de movimiento e imagen al robot. Al pulsar el campo "Puerto", aparece un nuevo recuadro donde poder incluir el puerto correspondiente, y también como en el caso anterior, un teclado táctil para poder introducirla, como se ve en la Figura 45. Como en el campo anterior, al iniciar la aplicación, viene por defecto un valor inicial, pero al modificarlo, se queda guardado el último para facilitar las posteriores conexiones con el robot.

Figura 45. Introducir puerto en preferencias.

- El campo "Cámara de Nao" es un *checkbox* que, al activarlo permite a la aplicación recibir las imágenes que el nodo *server* de ROS obtiene de la cámara del robot. Al desactivarlo se termina la recepción de imágenes. Como los campos anteriores, este también tiene por defecto el *checkbox* deseleccionado, pero se queda con el último valor que se le ha asignado.

- Las opciones de ajustes mencionadas anteriormente contienen una descripción breve y concisa de lo que es cada una.

- Al ser pulsado el botón "Ayuda" se despliega la ayuda de la aplicación (Figura 46), aquí se explica cómo realizar cada uno de los movimientos de las partes del cuerpo del Nao con los dos *joysticks*, para que un usuario inicial de la aplicación pueda basarse en ella, adaptarse a la interfaz y manejar correctamente los *joysticks*. Puesto que al ser una interfaz para dispositivos móviles, el tamaño de la pantalla es reducido, y no se pueden identificar como realizar cada uno de los movimientos que se llevan a cabo en el robot sobre la interfaz, debido a su complejidad.

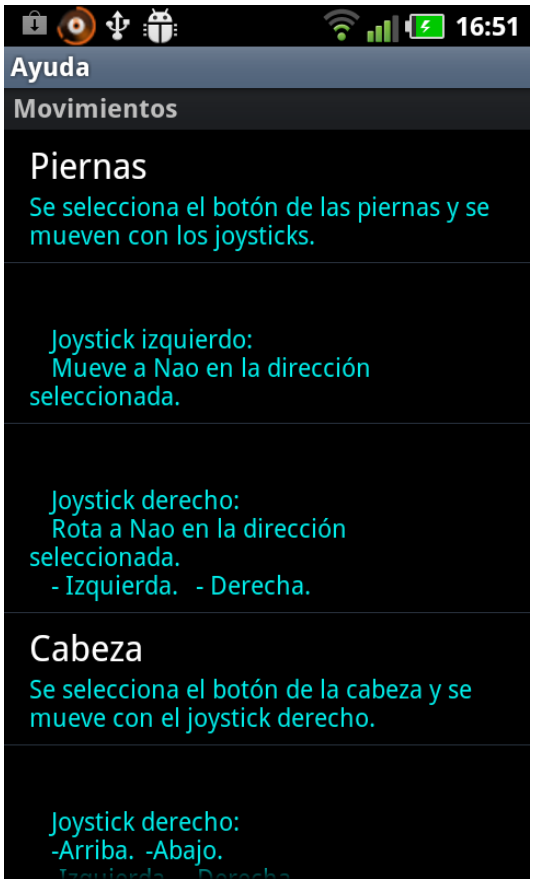

Figura 46. Ayuda de la aplicación.

- Se sale de la aplicación pulsando "Salir".

3.2.4.3 Rotación del dispositivo

- La aplicación se adapta al giro de pantalla de los dispositivos móviles, para poder utilizarla tanto en horizontal como en vertical, como se puede ver en la Figura 47.

Figura 47. Interfaz en horizontal.

El correcto uso de la aplicación, tanto de la conexión, como la selección y el movimiento de los botones y joysticks se explica en la sección del manual de usuario.

3.2.4.4 Limitaciones

A continuación se describen las limitaciones existentes en el sistema desarrollado:

- Aunque es compatible con todos los dispositivos Android, la imagen recibida del robot no aumenta si utilizas la aplicación en un *tablet*.
- La imagen recibida tiene un retraso debido al canal de comunicaciones y el tamaño de la misma.
- No es posible el cambio de cámara del robot a través de la aplicación.
- Solo se puede seleccionar el robot introduciendo su ip, en lugar de elegirlo de una lista de robots conectados a la misma red que el dispositivo móvil.
- La precisión que se consigue con la interfaz no es lo suficientemente buena para realizar tareas con objetos pequeños.
- La aplicación no permite levantar al robot si este está sentado o se cae.

3.2.4.5 Diagrama de flujo

En esta sección se detalla el diagrama de flujo del diseño final de la aplicación *NaoController*. Desde que se inicia la aplicación por el usuario, pasando por la configuración de la conexión, la visualización de la ayuda, la conexión con el nodo *server* y el envío y recepción de mensajes, tanto de movimiento, como de imagen. A continuación se presenta en la Figura 48 el diagrama de flujo completo de la aplicación.

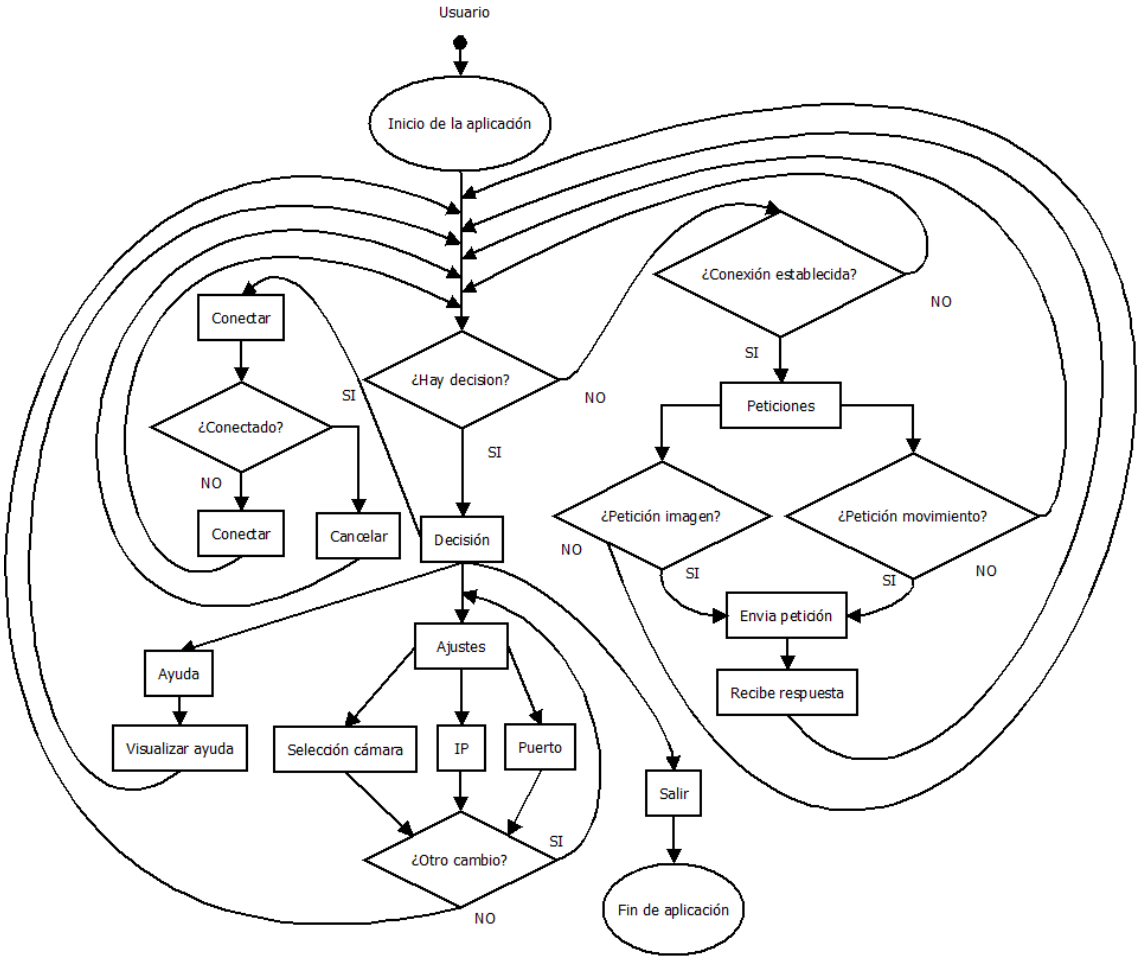

Figura 48. Diagrama de flujo NaoController.

Debido a la complejidad en su funcionamiento se divide en pequeños diagramas que van a ser descritos individualmente.

1. Al iniciar la aplicación en la funcionalidad “Inicio de la aplicación”, se comprueba si hay una decisión “¿Hay decisión?”, es decir, si el usuario ha elegido uno de los botones existentes en el submenú de la interfaz (Figura 49).

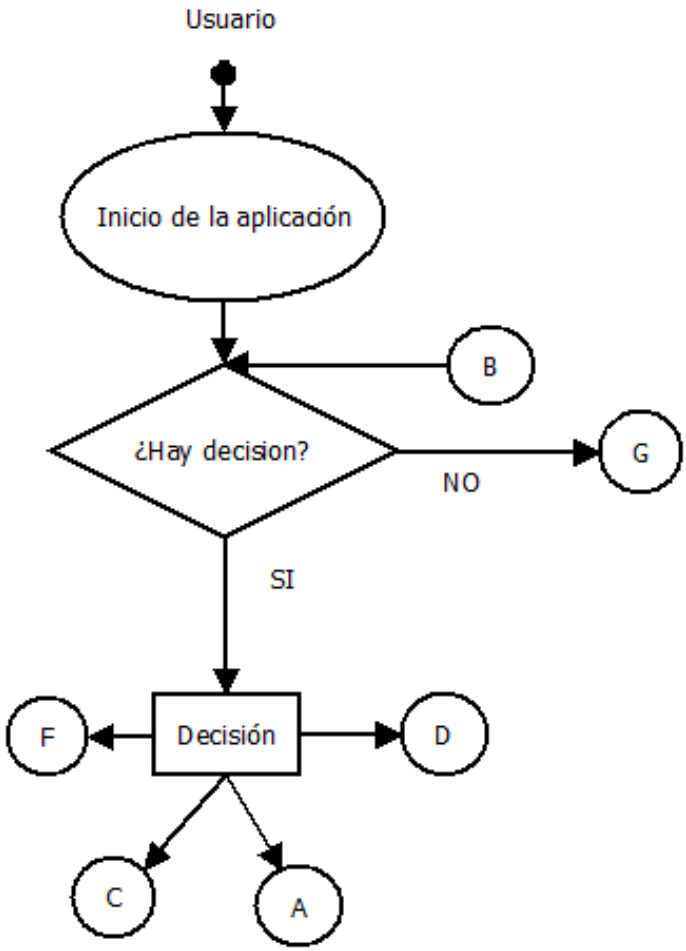

Figura 49. Diagrama de flujo NaoController, inicio de aplicación.

2. Si no ha elegido ningún botón del submenú, se comprueba que se haya establecido la conexión yendo por medio de “G” a “¿Conexión establecida?” en la Figura 50.

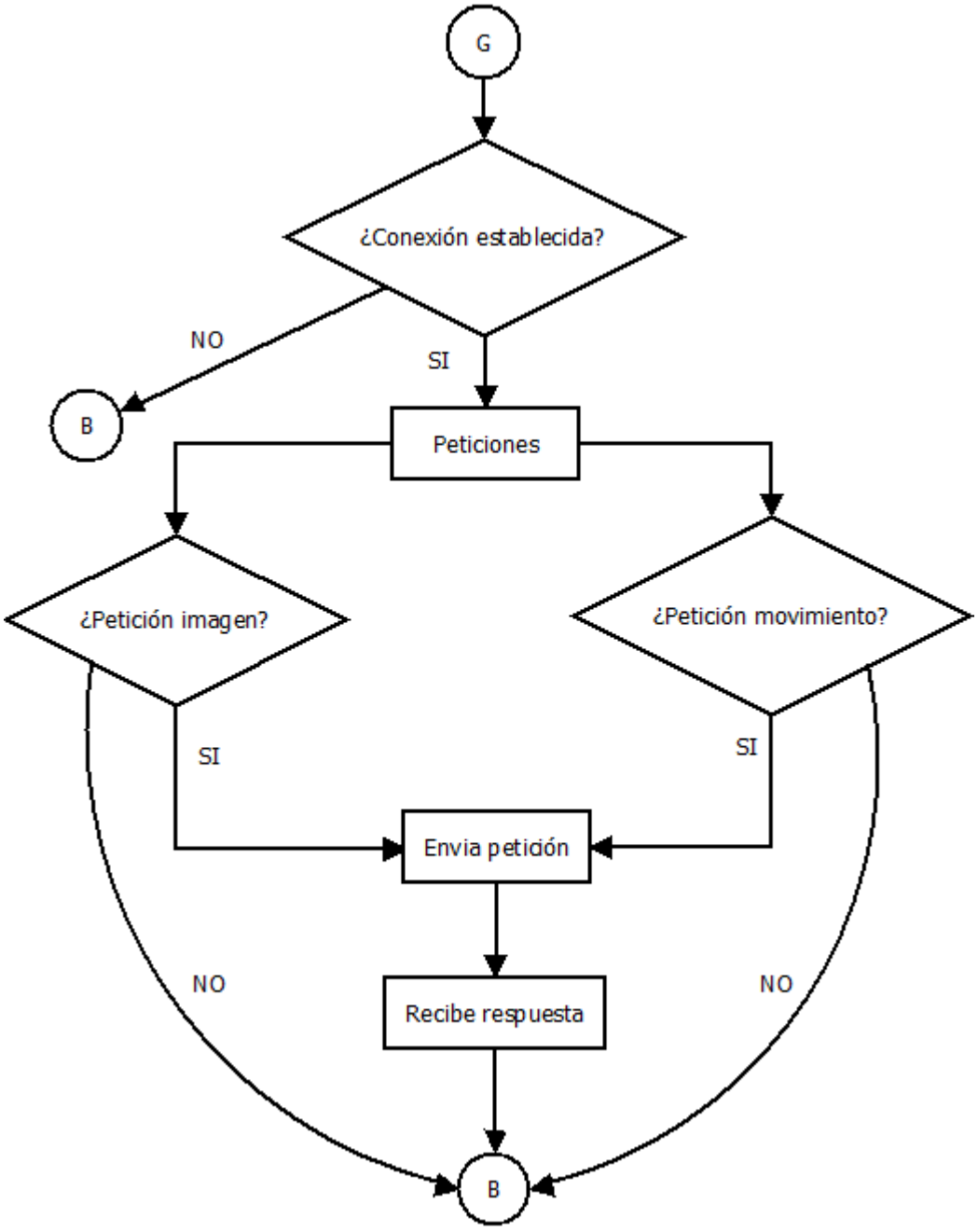

Figura 50. Diagrama de flujo NaoController, peticiones.

3. Si no se ha establecido la conexión, se vuelve a comprobar que haya una decisión por parte del usuario "¿Hay decisión?" (Figura 49) volviendo al punto 1.

4. Si hay conexión establecida, se comprueba que haya peticiones "Peticiones", de imagen "¿Petición de imagen?" o de movimiento "¿Petición movimiento?".

    - La petición de imagen proviene de la selección por parte del usuario de la recepción de imágenes en el menú de preferencias de conexión de la aplicación (Figura 50).

    - La petición de movimiento surge como resultado de mover los *joysticks* existentes en la interfaz, para mover una de las partes del cuerpo del robot (Figura 50).

5. En caso afirmativo, envía la petición por medio de la funcionalidad "Envía petición".

6. Recibe la respuesta pertinente con la funcionalidad "Recibe respuesta" y pasa a comprobar si nuevamente hay decisión "¿Hay decisión?" en el apartado 1, si no hay petición, directamente pasa a comprobar si existe decisión "¿Hay decisión?" en el apartado 1 (Figura 49 y Figura 50).

7. Cuando se realiza alguna decisión en el submenú, se comprueba el tipo de decisión "Decisión" (Figura 49).

8. Si se ha seleccionado el botón “Conectar”.

    - Si la aplicación está conectada “¿Conectado?”, cancela la conexión por medio de la funcionalidad “Cancelar” y vuelve a esperar una decisión “¿Hay decisión?” en el apartado 1.

    - Si por el contrario, no está conectada, la aplicación se conecta “Conectar”, y vuelve a esperar una decisión “¿Hay decisión?” en el apartado 1, o petición de imagen “¿Petición imagen?” o movimiento “¿Petición movimiento?” en el apartado 4 (Figura 49, Figura 50 y Figura 51).

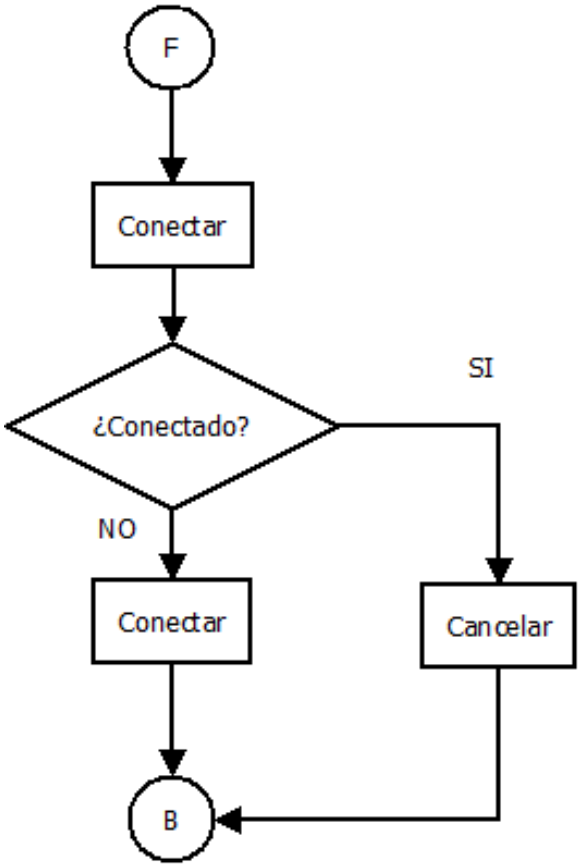

Figura 51. Diagrama de flujo NaoController, conexión.

9. Si selecciona “Ayuda”, el usuario visualiza la ayuda “Visualizar ayuda” el tiempo que le sea necesario, y posteriormente, se vuelve a esperar una petición de imagen “¿Petición imagen?” o movimiento “¿Petición movimiento?” en el apartado 4, o una nueva decisión “¿Hay decisión?” en el 1 (Figura 49, Figura 50 y Figura 52).

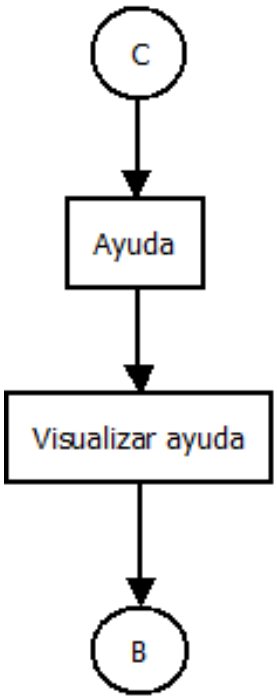

Figura 52. Diagrama de flujo NaoController, ayuda

10. Si se ha seleccionado “Ajustes”.

    - Se guarda el cambio que el usuario haya introducido (en “IP”, “Puerto” o “Selección cámara”).

    - Se comprueba si hay otro cambio “¿Otro cambio?”, si lo hay, se guarda nuevamente, si no, se vuelve a esperar una decisión “¿Hay decisión?” en el apartado 1, o una petición de imagen “¿Petición imagen?” o movimiento “¿Petición movimiento?” en el 4 (Figura 49, Figura 50 y Figura 53).

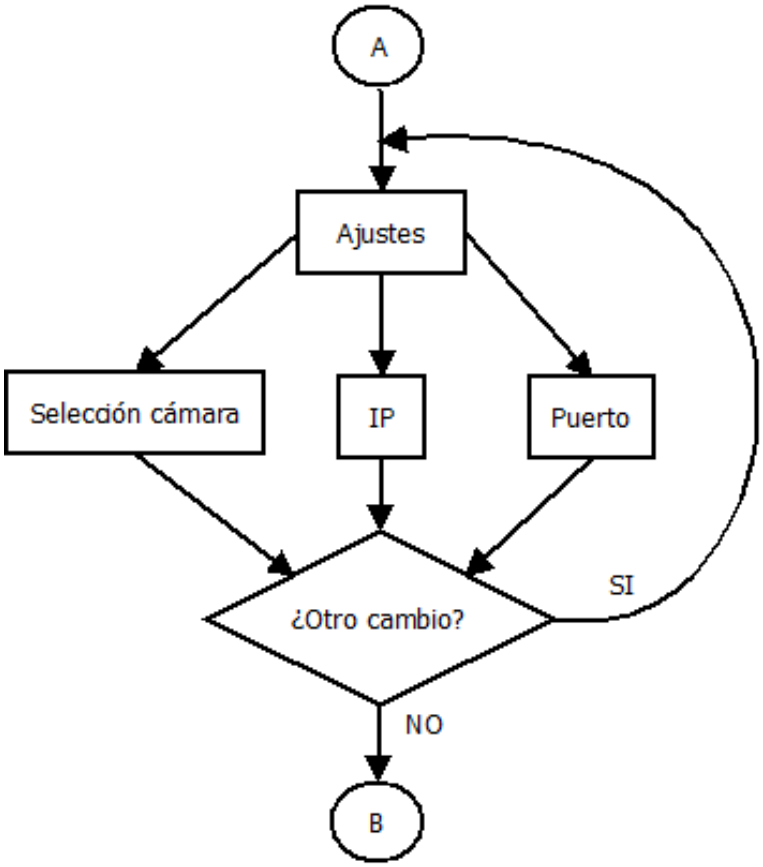

Figura 53. Diagrama de flujo NaoController, ajustes.

11. Si la decisión tomada es “Salir”, el usuario se sale de la aplicación, finaliza la conexión y se cierra la aplicación “Fin de aplicación” (Figura 54).

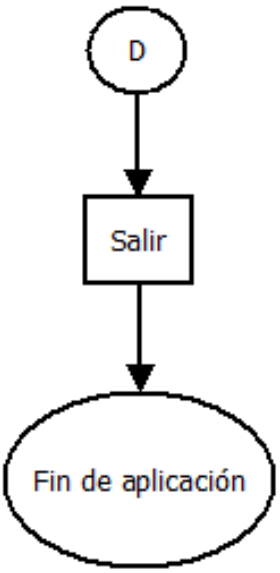

Figura 54. Diagrama de flujo NaoController, fin de aplicación.

Capítulo 4

Pruebas preliminares y evaluación

Antes de realizar una evaluación de la aplicación se llevaron a cabo una serie de pruebas preliminares para comprobar el correcto funcionamiento de la misma y el servidor, y comprobar que no surgía ningún error.

4.1 Pruebas preliminares

A continuación se realiza una descripción de las pruebas preliminares que fueron realizadas sobre la aplicación, así como el resultado esperado y el resultado obtenido. Cada una se llevó a cabo en el siguiente entorno operacional:

- Un entorno real y otro simulado.
- El robot humanoide NAO.
- Un dispositivo móvil android en el que estaba el módulo de teleoperación.
- Un ordenador en el que se encontraba el módulo de control.
- Todas ellas realizadas con y sin recepción de imagen en la aplicación.

- Estableciendo la comunicación entre los distintos dispositivos por medio de una red LAN, a través de un router inalámbrico.

4.1.1 Conexión

Se llevó a cabo la conexión desde la aplicación *NaoController* al nodo servidor de ROS. Con el servidor en ejecución se lanzó la aplicación y se introdujeron los parámetros correctos de conexión para establecerla, posteriormente se pulsó el botón de conexión en la aplicación. El resultado esperado era la conexión con el módulo de control y la recepción de imágenes, en el caso de que estuviera activada. El resultado obtenido fue la correcta conexión de la aplicación, y la visualización de las imágenes captadas por el robot, en el caso de la recepción de imágenes activadas.

La prueba se realizó tanto en el simulador como en el dispositivo real, siendo en ambos casos satisfactoria.

Una demostración de esta prueba se puede ver en el siguiente enlace:

http://www.youtube.com/watch?v=vlvMfgK-EKY&feature=plcp

4.1.2 Evasión de obstáculos

Con la aplicación conectada, se desplazó al robot para esquivar una serie de obstáculos Figura 55. El resultado esperado era el desplazamiento del robot entre los obstáculos sin chocar con ninguno. El resultado obtenido fue el esperado, se consiguió desplazar al robot entre los obstáculos, tanto desplazándose hacia cualquier dirección, como girando sobre sí mismo.

Figura 55. Evasión de obstáculos.

En el caso de la prueba utilizando la recepción de imágenes, se realizó la teleoperación del Nao con solo la visualización de las mismas. Comprobando así el funcionamiento del movimiento de la cabeza, ya que para esquivar los obstáculos había que moverla en función del desplazamiento que este hiciera, y de esa forma ver hacia donde se estaba moviendo. El resultado esperado era el paso entre los obstáculos sin chocar, y la visualización de la dirección de desplazamiento por medio del movimiento de la cabeza. Se consiguió esquivar los obstáculos, pero en un tiempo mayor, debido al movimiento de la cabeza y al menor grado de visión ofrecido. Esta prueba solo se realizó en el entorno real.

Una demostración de esta prueba se puede ver en el siguiente enlace:

http://www.youtube.com/watch?v=s7u02VmXHEw&feature=plcp

4.1.3 Interacción con objetos

Se probó el correcto funcionamiento del movimiento de las articulaciones de los brazos, este es el movimiento más complejo existente en la aplicación. Se utilizó un objeto de gomaespuma, que se le dió al robot para que lo cogiera por medio de la apertura de la mano, y lo soltara encima de una superficie (Figura 56).

Figura 56. Dejar un objeto.

Por cada articulación que se quería mover, se miraba la ayuda para ver cuál era el uso correcto de los controles de la aplicación. El resultado esperado era dejar correctamente el objeto encima de la superficie. El resultado obtenido fue el posicionamiento del objeto sobre la superficie.

También se usó la recepción de imágenes para comprobar que se podía realizar por medio de las mismas. Se utilizó el movimiento de la cabeza para ver correctamente donde se estaba situando el objeto. El resultado esperado era el posicionamiento del objeto sobre la superficie. El resultado obtenido fue el esperado. Se tardó un mayor tiempo debido a los mismos motivos anteriormente mencionados en el apartado 4.1.2. Solo se realizó esta prueba en el entorno real.

Una demostración de esta prueba se puede ver en el siguiente enlace:

http://www.youtube.com/watch?v=P3vg3VWrozA&feature=plcp

4.2 Evaluación

A continuación se expone la evaluación del sistema de teleoperación desarrollado en este trabajo. Se escogieron a seis usuarios que por medio de la interfaz de la aplicación, se les propuso realizar una serie de tareas preestablecidas para comprobar el funcionamiento de la misma. Son tareas típicas que se tendrían que realizar para poder teleoperar correctamente el robot Nao. Como objetivos a medir en esta evaluación, se tenían los siguientes:

- Usabilidad de la aplicación. Con esto se quiere comprobar si el usuario encuentra fácil e intuitivo el uso de los botones, controles y en general, el uso de la aplicación en su conjunto.

- Averiguar si el usuario sabe utilizar correctamente los controles desarrollados.

- Saber si la recepción de imágenes es útil para la teleoperación de la aplicación.

- Comprobar si el usuario puede alcanzar los objetivos determinados en las tareas propuestas.

- Comprobar si el usuario se encuentra satisfecho y encuentra de utilidad la aplicación como método de teleoperación de robots.

- Conocer la opinión del usuario acerca de posibles mejoras que se podrían llevar a cabo, o sugerencias que ayudarían a mejorar la aplicación.

Esta evaluación se realizó de manera individualizada, explicándole al usuario, si no las entendía, las tareas descritas, y ayudándolo en el caso de que no estuviera familiarizado con las tecnologías utilizadas o lo necesitara.

Para mejorar la posterior recogida de los datos y facilitar la tarea a los usuarios, el cuestionario se realizó por medio de google docs, donde te permiten hacer cuestionarios de cualquier tipo y posteriormente descargarte el resultado en un excell, incluso mostrando gráficas y datos estadísticos. A continuación se exponen las pruebas y el cuestionario para realizar la evaluación.

Nombre: Fecha:

El proceso de evaluación de la aplicación NaoController estará formado por varias tareas. Estas permitirán al usuario analizar el funcionamiento de la aplicación móvil mediante el movimiento de las distintas partes del cuerpo del robot (cabeza, brazos y piernas), la recepción de imágenes por medio de la cámara del robot, andar para esquivar un obstáculo y dejar un objeto. Las tareas a realizar son las siguientes.

1. En primer lugar el usuario debe establecer una conexión con uno de los robots a través del menú inferior de la aplicación NaoController, para ellos se le proporcionará una dirección ip y un puerto.
2. Una vez establecida la comunicación con el robot, el usuario deberá permitir la recepción de imágenes en tiempo real.
3. Después de establecer la conexión visual con el robot, el usuario deberá comprobar el correcto funcionamiento del sistema de teleoperación ofrecido por la aplicación.
    a. El usuario deberá mover el robot desde una posición inicial a una final evadiendo un obstáculo, para ello debe hacer uso de la cabeza para ver el obstáculo y moverse, viendo el escenario solo a través de la recepción de imágenes.
    b. Posteriormente el usuario intentará acercarse lo más posible a una superficie, se le dará un objeto al robot e intentará soltarlo por medio de los brazos. Para ello también deberá hacer uso de la cabeza para visualizar el objeto.
    Se puede hacer uso del botón de ayuda situado en el menú de configuración inferior de NaoController.

Para evaluar las pruebas descritas, se realiza el siguiente cuestionario.

Tarea 1:

Configuración de la conexión con el robot

1. ¿Ha sido fácil realizar la conexión con el robot?

Sí ☐ No ☐

En caso negativo indique porqué.

2. ¿Se explicaba con claridad la información a introducir en cada campo?

Sí ☐ No ☐

En caso negativo indique porqué.

3. ¿Le ha resultado intuitivo el interfaz de conexión?

Sí ☐ No ☐

En caso negativo indique porqué.

Tarea 2:

Configuración de la recepción de imágenes

4. ¿Le ha resultado fácil activar la recepción de imágenes?

Sí ☐ No ☐

En caso negativo indique porqué.

5. ¿En la descripción que había en el botón para configurar la recepción de imágenes se explicaba con claridad lo que se conseguía activando las imágenes?

Sí ☐ No ☐

6. ¿Le ha resultado intuitivo el interfaz para poder recibir las imágenes?

Sí ☐ No ☐

En caso negativo indique porqué.

Tarea 3:

Evasión de obstáculos

7. ¿Ha sido fácil seleccionar la opción para mover el robot?

Sí ☐ No ☐

8. ¿El movimiento del robot se adaptaba a lo que se realizaba por medio del interfaz?

Sí ☐ No ☐

9. ¿Le ha resultado fácil esquivar los objetos?

Sí ☐ No ☐

En caso negativo indique porqué.

Movimiento de la cabeza

10. ¿Ha resultado fácil seleccionar la opción de mover la cabeza?

Sí ☐ No ☐

11. ¿Los movimientos seleccionados se adaptaban a lo que se realizaba por medio del interfaz?

Sí ☐ No ☐

12. ¿La imagen recibida en la aplicación al mover la cabeza se adecuaba al movimiento realizado con un retraso aceptable?

Sí ☐ No ☐

Movimiento de los brazos

13. ¿Le ha resultado fácil adaptarse al control de los dos *joysticks* para mover las articulaciones de los brazos?

Sí ☐ No ☐

En caso negativo indique porqué.

14. ¿Le ha parecido intuitivo?

Sí ☐ No ☐

En caso negativo indique porqué.

15. ¿Se adaptaba el movimiento del robot al movimiento realizado por medio del interfaz?

Sí ☐ No ☐

Interacción y recogida de objetos

16. ¿Ha podido cambiar fácilmente entre los botones de las partes del cuerpo del robot?

Sí ☐ No ☐

En caso negativo indique porqué.

17. ¿Le ha parecido fácil dejar al robot a una distancia cercana de la superficie donde se encontraba el objeto a dejar?

Sí ☐ No ☐

En caso negativo indique porqué.

18. ¿Le ha resultado fácil dejar el objeto?

Sí ☐ No ☐

En caso negativo indique porqué.

Preguntas generales:

19. ¿Le ha resultado intuitivo el interfaz con la que se realizan los movimientos en el robot?

Sí ☐ No ☐

En caso negativo indique porqué.

20. ¿En general le han resultado precisos los movimientos realizados con el robot?

Sí ☐ No ☐

En caso negativo indique porqué.

21. ¿Ha tenido que utilizar la ayuda de la aplicación?

Sí ☐ No ☐

En caso afirmativo, indica porqué.

22. ¿Ha necesitado asistencia para que le indicaran como realizar alguna de las tareas?

Sí ☐ No ☐

En caso afirmativo indica en cual y porqué.

23. ¿Le parece útil la aplicación como base para posteriores mejoras y poder utilizarla con otro tipo de robots?

Sí ☐ No ☐

En caso negativo, indica porqué.

24. ¿Le parece útil la aplicación como base para posteriores mejoras?

Sí ☐ No ☐

En caso negativo, indica porqué.

25. ¿Cree que sería buena idea poder utilizarla con otro tipo de robots?

Sí ☐ No ☐

En caso afirmativo o negativo, indica porqué.

26. ¿Cree que esta aplicación podría ser utilizada en la teleoperación de un robot para realizar tareas peligrosas para el ser humano, en entornos de difícil acceso o para asistencia personal?

Sí ☐ No ☐

En caso afirmativo o negativo, indica porqué.

27. ¿Qué aspectos mejoraría de la aplicación?

Otras observaciones

A continuación, en la Tabla 65, se presentan las respuestas obtenidas al realizar el cuestionario anterior a los usuarios que probaron el sistema.

Pregunta	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5	Usuario 6
1	Si	Si	Si	Si	Si	Si
2	Si	Si	Si	Si	Si	Si
3	Si	Si	Si	Si	Si	No
4	Si	Si	Si	Si	Si	Si
5	Si	Si	Si	Si	Si	Si
6	Si	Si	Si	Si	Si	Si
7	Si	Si	Si	No	Si	No
8	Si	Si	Si	Si	Si	No
9	Si	Si	Si	Si	No	Si
10	Si	Si	Si	No	Si	Si
11	Si	Si	Si	Si	Si	Si
12	Si	No	Si	Si	No	Si
13	No	Si	Si	Si	Si	No
14	No	Si	Si	No	Si	No
15	Si	Si	Si	Si	Si	Si
16	Si	Si	Si	No	Si	Si
17	Si	Si	Si	Si	Si	Si
18	Si	Si	Si	No	Si	Si
19	No	Si	Si	No	Si	No
20	Si	Si	Si	Si	Si	Si
21	No	Si	No	No	Si	No
22	No	Si	No	Si	Si	Si
23	Si	Si	Si	Si	Si	Si
24	Si	Si	Si	Si	Si	Si
25	Si	Si	Si	Si	Si	Si
26	Si	No	Si	No	Si	Si

Tabla 65. Respuestas al cuestionario de evaluación.

Tras la realización del cuestionario, en términos generales, se han obtenido las siguientes conclusiones:

- Todos los usuarios estaban de acuerdo en que la conexión con el robot se realizaba fácilmente. Así como los ajustes de la conexión y las breves descripciones en ellas, que describían correctamente el uso que se les daba, y resultaba sencillo e intuitivo introducir los parámetros de conexión o activar la recepción de imágenes.

- También están de acuerdo en que es fácil cambiar entre las distintas partes del cuerpo del robot en la interfaz por medio de los botones creados para ello.

- Les resultó fácil mover el robot para esquivar el obstáculo, adaptándose el movimiento a lo realizado sobre la interfaz. Al realizar el movimiento mirando solo desde la interfaz a través de la recepción de imágenes, a algunos les costó más que a otros, alegando que el refresco de la imagen es en algunos casos lento.

- Al acercar con precisión el robot a la superficie para dejar el objeto, resultó fácil en todos los casos, ya que la precisión que hay que realizar para mover las piernas, es menor que la que se necesita para mover los brazos.

- Al realizar el movimiento de los brazos para mover las articulaciones y dejar el objeto, en general resultó fácil adaptarse a los controles, aunque hubo algunos problemas para cambiar entre las distintas articulaciones al principio. El movimiento del Nao se adaptaba a los movimientos realizados con la interfaz, pero al ser los *joysticks* pequeños, resultaba un poco difícil realizarlo con mucha precisión.

- En general los usuarios necesitaron el uso de la ayuda o asistencia, pero solo al principio para adaptarse a los movimientos de la interfaz. Posteriormente, tras utilizar la interfaz durante unos minutos, en general los usuarios se adaptaron bien a los controles y botones, confirmando que es fácil de manejar.

- Se cree que podría ser útil para utilizarla como base en la teleoperación de otros robots, adaptándola dependiendo del tipo de robot. Es una aplicación sencilla adaptable y menos costosa que si se utilizara un dispositivo especializado.

- Se podría utilizar para realizar tareas peligrosas pero habría que mejorar la precisión, incorporar algunos mensajes de ayuda, como por ejemplo que es lo que estás moviendo en cada momento y mantenerlo de manera fija, en el caso de las articulaciones. Aumentar la precisión en el movimiento de los brazos, y la velocidad en la transmisión de imágenes.

- Algunos de los usuarios probaron la aplicación además de por wifi, con el cable de red conectado al robot, y la imagen llegaba más rápido a la aplicación, mejorando notablemente la interacción con el robot.

Capítulo 5

Gestión del trabajo

En este capítulo se describen cada una de las fases que han formado parte del proceso de desarrollo de este trabajo, junto con cada una de las tareas que las conforman. Se incluye una descripción de los medios necesarios empleados en el desarrollo del trabajo, un presupuesto en el que se indican los costes derivados del material y el personal necesario, así como un diagrama Gantt en el que se representan gráficamente cada una de las fases, tareas y relaciones existentes entre ellas.

5.1 Fases del desarrollo

El trabajo se ha dividido en varias fases, siguiendo un ciclo de vida en espiral, modificándose progresivamente para poder avanzar en la creación del sistema de teleoperación.

- Análisis de la tecnología. Se analizan los sistemas operativos que se van a utilizar, como son Android y ROS. Así como, el resto de *software* (el entorno de programación Eclipse, el kit de desarrollo JDK, el entorno de simulación Choregraphe, el *middleware* NaoQi y el paquete de funcionalidades de la Universidad de Friburgo).

- Toma de requisitos. Se definen las funcionalidades, restricciones y casos de uso, en los que se basa el sistema de teleoperación.

- Aprendizaje. Se adquieren los conocimientos necesarios para programar en Android, C++ y utilizar ROS.

- Diseño. Se diseñan los módulos de control y teleoperación que se van a desarrollar posteriormente en el sistema de teleoperación.

- Implementación. Se desarrollan los módulos de control y teleoperación que conforman el sistema de teleoperación.

- Pruebas. Se realizan las pruebas que se van a ejecutar con el sistema de teleoperación y se genera un informe de estas.

- Documentación. Se elabora la memoria que describe el trabajo realizado y se preparan su presentación.

5.2 Distribución de tareas

En la Tabla 66 se presentan el conjunto de fases y tareas realizadas durante este trabajo, mostrando para cada una, el nombre, el número de días requeridos, la fecha de inicio y la de finalización. Aunque no ha habido periodo vacacional durante la realización de este trabajo, ha habido algunos días de descanso, por lo que todas las fechas no son consecutivas.

La planificación de este trabajo se puede dividir en: Análisis de la tecnología, Toma de requisitos, Aprendizaje, Diseño, Implementación, Pruebas y Documentación. En la Figura 57 se presenta el diagrama de Gantt realizado.

Nº	Tarea	Duración	Inicio	Fin
1	Análisis de la tecnología	19 días	22/03/2012	09/04/2012
2	Análisis de Android	5 días	22/03/2012	26/03/2012
3	Análisis de ROS	9 días	27/03/2012	04/04/2012
4	Análisis del resto de software	5 días	05/04/2012	09/04/2012
5	Toma de requisitos	8 días	11/04/2012	18/04/2012
6	Definición de funcionalidades	2 días	11/04/2012	12/04/2012
7	Definición de restricciones	2 días	13/04/2012	14/04/2012
8	Definición casos de uso	4 días	15/04/2012	18/04/2012
9	Aprendizaje	26 días	21/04/2012	16/05/2012

Nº	Tarea	Duración	Inicio	Fin
10	Aprendizaje ROS	12 días	21/04/2012	02/05/2012
11	Aprendizaje Android	8 días	03/05/2012	10/05/2012
12	Aprendizaje C++	6 días	11/05/2012	16/05/2012
13	Diseño	18 días	18/05/2012	05/06/2012
14	Interfaz Usuario-Controlador (*NaoController* y nodo servidor de ROS)	9 días	18/05/2012	26/05/2012
15	Nodo servidor ROS-NAO	9 días	28/05/2012	05/06/2012
16	Implementación	44 días	08/06/2012	21/07/2012
17	Interfaz Usuario-Controlador (*NaoController* y nodo servidor de ROS)	23 días	08/06/2012	30/06/2012
18	Nodo servidor ROS-NAO	19 días	3/07/2012	21/07/2012
19	Pruebas	14 días	22/07/2012	04/08/2012
20	Realización de pruebas	9 días	22/07/2012	30/07/2012
21	Informe de pruebas	5 días	31/07/2012	04/08/2012
22	Documentación	28 días	07/08/2012	03/09/2012
23	Elaboración de memoria	25 días	07/08/2012	31/08/2012
24	Elaboración y preparación de presentación	3 días	1/09/2012	03/09/2012

Tabla 66. División del trabajo en tareas.

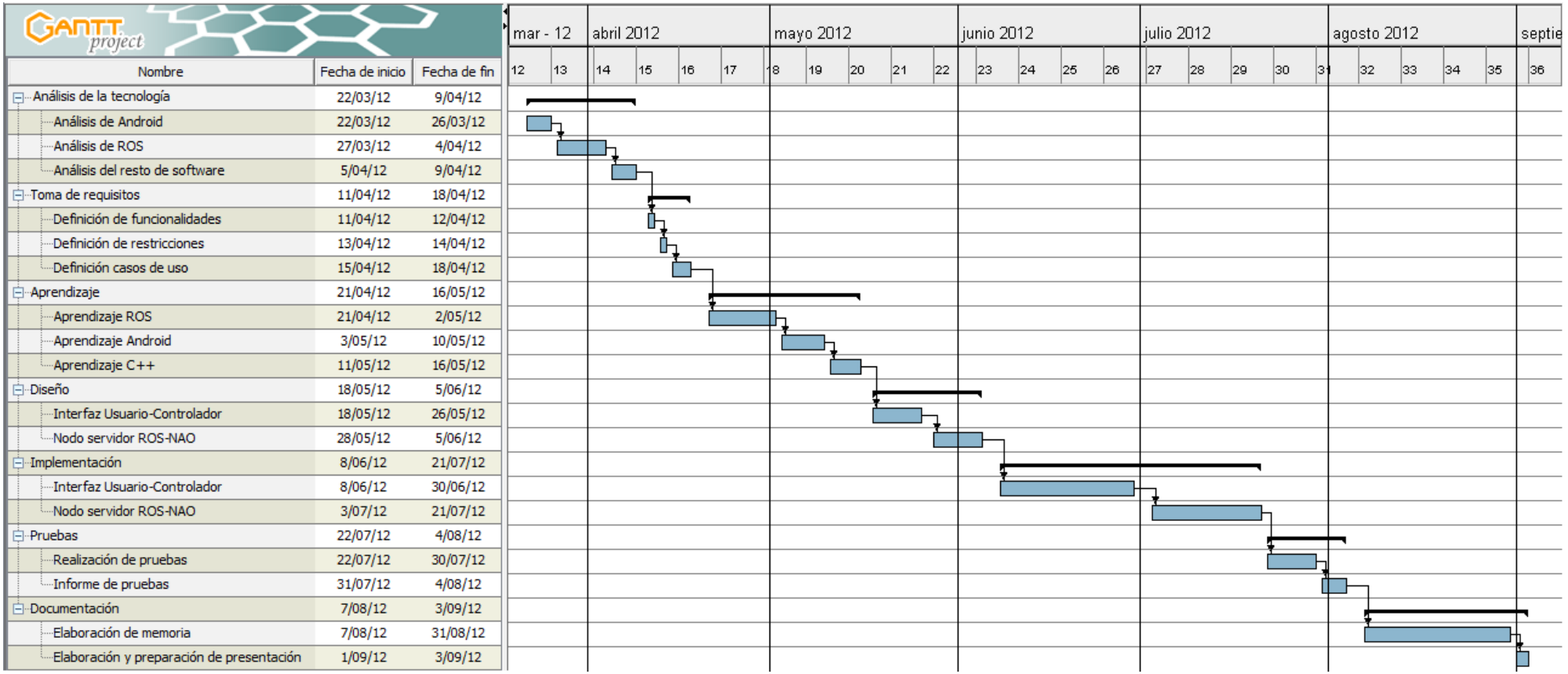

Figura 57. Diagrama de Gantt.

5.3 Medios empleados

Para la realización de este trabajo se ha contado con los siguientes medios:

- Un ordenador portátil Asus K53S, con el sistema operativo Ubuntu 10.04, no se han cogido versiones superiores por problemas de compatibilidad, en donde se han llevado a cabo las tareas de programación, tanto de la aplicación en android como del servidor para el NAO.

- Un móvil LG Optimus 3D, con el sistema operativo Android 2.3 Gingerbread, en el que se ha implementado la interfaz gráfica para que el usuario pueda teleoperar el robot por medio de la aplicación creada.

- El router Xavi 7869, a través del cual se ha podido realizar la comunicación y transmisión de datos entre la aplicación en Android y el Nao.

- Para la realización de las pruebas de la aplicación, se ha utilizado un emulador de un dispositivo con sistema operativo android 2.1 Eclair, que se hace posible gracias al android-sdk-r18, kit de desarrollo de *software* que provee ciertos paquetes, entre los cuales, lo trae como una de sus herramientas. Se ha utilizado una versión inferior que en el dispositivo real para aumentar la compatibilidad con dispositivos anteriores.

- El entorno de programación eclipse, es una herramienta para facilitar la programación con herramientas como la de autocompletado, tanto para la aplicación en Android programada en Java, como para el servidor del robot, programado en C++.

- El kit de desarrollo de Java (JDK), que es un *software* para la creación de programas en java, necesario para llevar a cabo la aplicación, puesto que android está basado en java.

- ROS, un sistema operativo de código abierto para robots, proporciona las herramientas y bibliotecas necesarias para la construcción, escritura y ejecución de código realizado en este trabajo.

- El paquete de funcionalidades de la Universidad de Friburgo para ROS, que aporta un conjunto de librerías facilitando la programación del servidor para el robot humanoide NAO.

- El *middleware* de programación NaoQi, para el desarrollo de aplicaciones en el robot Nao, facilitando el acceso a los sensores (cámaras, ultrasonidos, sensores de contacto,…) y actuadores (motores, leds,…) para el desarrollo del servidor en C++. Permitiendo ejecutarse tanto en el robot real como en un ordenador en un entorno de simulación.

- El entorno de programación y simulación Choregraphe, que ha permitido probar el servidor a la vez que se ha ido creando, y simularlo en un robot virtual de manera simple y sencilla.
- Robot humanoide NAO, al que se le envían las órdenes necesarias desde el servidor, que han sido enviadas anteriormente desde la aplicación realizada en Android, para interpretarlas y actuar en consecuencia.

5.4 Presupuesto

Se presenta a continuación el presupuesto y su desglose; personal, seguridad social, medios materiales y amortización del mismo, así como el costo total del trabajo.

5.4.1 Cálculo de coste de personal con seguridad social.

Se abonan dos pagas extraordinarias al año, las cuales no incluyen seguridad social, por lo que hay calcular el coste de la seguridad social en el resto de los meses del año. Aplicamos la siguiente fórmula, se calcula el término C:

C = (A / B) + A

A = Total cantidad pagada al trabajador.
B = 6, es el número de meses en los que se aplica la seguridad social de una paga extraordinaria.

A continuación la base reguladora E, quitándole los decimales para redondear mediante la siguiente fórmula:

E = C / D

C = Término anteriormente calculado.
D = 300

Se multiplica D * E, y se obtiene la cifra redondeada F.

Los conceptos por los que paga la empresa de seguridad social son:

Contingencias comunes → 23.6%
Accidentes de trabajo → 2%
Desempleo → 5.5%
Fondo garantía salarial → 0.2%
Formación profesional → 0.6%
Total → 31.9%

Total pagado por empleado = F * 31.9 /100

5.4.2 Cálculo de amortización de recursos

(A/B)*C

A = nº de meses que el equipo es utilizado.
B = período de depreciación (48 meses).
C = coste del equipo.

5.4.3 Cálculo de presupuesto

Autor:	
Juan Domingo Gálvez Cobo	
Departamento:	
Informática	
Descripción del trabajo:	
Título:	Teleoperación del robot NAO mediante dispositivos móviles Android.
Duración:	8 meses

Tabla 67. Datos del trabajo.

El precio/mes, se obtiene de las tablas salariales del BOE, nº 174 del 21 de julio de 2012.

Personal				
	Precio/mes (euros)	**Número meses**	**Coste final (euros)**	**Coste final con seguridad social (euros)**
Jefe de proyecto	1885.17	1.50	2827.75	3880.45
Analista	1536.07	0.86	1321.02	1799.52
Investigador	1581.59	1.50	2372.38	3233.68
Programador	1187.59	2.86	3404.42	4648.52
Total personal				13562.17

Tabla 68. Costes de personal.

Recursos				
Recurso	**Precio (euros)**	**Dedicación (meses)**	**Período de depreciación (meses)**	**Amortización (euros)**
Asus K53S	699.00	8	48	116.50
LG Optimus 3D	299.00	7	48	43.60
Xavi 7869	30.00	7	48	4.37
Ubuntu 10.04	0.00	7	0	0.00
Android-sdk-r18	0.00	7	0	0.00
Eclipse	0.00	7	0	0.00
JDK	0.00	7	0	0.00
ROS	0.00	7	0	0.00
NaoQi	0.00	7	0	0.00
Choregraphe	0.00	7	0	0.00
Nao	12821.33	7	48	1869.77
Total amortización				2034.24

Tabla 69. Recursos.

Resumen de costes	
Personal	13562.17 €
Amortización	2034.24 €
Subcontratación de tareas	0.00 €
Costes de funcionamiento	0.00 €
Presupuesto	15596.41 €
Costes indirectos (10%)	1559.64 €
Presupuesto ejecución material	17156.05 €

Tabla 70. Resumen de costes.

"El presupuesto total de este trabajo asciende a la cantidad de DIECISIETE MIL CIENTO CINCUENTA Y SEIS CON CERO CINCO EUROS (17156.05 €).

Leganés a X de Agosto de 2012

El ingeniero proyectista

Fdo. Juan Domingo Gálvez Cobo

Capítulo 6

Conclusiones y líneas futuras

En este capítulo se exponen las conclusiones obtenidas a lo largo de la realización del trabajo, describiendo las dificultades encontradas en su desarrollo. Primero se presentan las conclusiones generales extraídas al finalizar el trabajo, posteriormente se describen las referentes a los objetivos marcados al comienzo del mismo y los problemas encontrados, y por último las líneas futuras que se podrían implementar sobre el sistema de teleoperación.

6.1 Conclusiones generales

Durante el desarrollo de este trabajo, he adquirido una serie de conocimientos orientados a la creación de sistemas de control por medio de la teleoperación.

El aprendizaje de nuevos lenguajes de programación, como Android, basado en Java, orientado al desarrollo de aplicaciones para dispositivos móviles.

El uso de métodos de simulación utilizando *software*, como el proporcionado por Aldebaran Robotics, Choregraphe. Y otras técnicas para poder simular elementos, como la cámara del robot, cuando este no estaba disponible.

La utilización de sistemas especializados en el control y desarrollo de sistemas de control de robots, como ROS y el uso de *middleware*, como NaoQi.

También me ha permitido tener una visión general más realista acerca de lo que es la teleoperación, utilizando dispositivos con Android, y la creación de los sistemas de control para agentes físicos. Comprobar que se puede usar la tecnología actual que está a disposición de cualquiera, como son los dispositivos móviles con Android, y utilizarla para adaptarla a la creación de este tipo de sistemas de teleoperación, de una manera simple e intuitiva.

Conocer el trabajo en un laboratorio de investigación utilizando los medios materiales disponibles, que no se utilizan normalmente durante la carrera.

6.2 Conclusiones referentes a los objetivos

En este apartado se realiza un análisis de los objetivos planteados al inicio del trabajo, y su cumplimentación.

1. Estudio y análisis de ROS (*Robot Operating System*).
   Se ha conseguido comprender el funcionamiento de ROS durante el desarrollo de este trabajo, asimilando cómo funciona cada una de las funcionalidades utilizadas. Gracias al estudio previo y la realización de algunos tutoriales, ha sido posible un mejor diseño y posterior implementación, ya que si no, hubiera sido necesario rehacer gran parte de los mismos. También cabe destacar que, aun habiendo información relevante, en algunos casos no era suficiente, y ha sido necesaria la investigación observando proyectos similares y la búsqueda en foros. Hay una gran comunidad, pero no está lo suficientemente expandida.

2. Estudio y análisis del sistema operativo Android.
   Tras asimilar los conceptos y conocimientos referentes a las funcionalidades y librerías específicas de Android, estando este basado en Java, ha sido posible el posterior diseño e implementación de una interfaz intuitiva y sencilla. Cabe destacar la gran cantidad de información existente en la red.

3. Diseño e implementación del sistema de teleoperación.
   Ha sido la fase más compleja y a la vez la más satisfactoria de todo el trabajo, ya que me ha permitido utilizar dispositivos con los que habitualmente no he trabajado, como son los robots y los dispositivos móviles, centrarme en el desarrollo de un sistema de teleoperación partiendo de cero, y utilizar entornos de simulación para probar los avances en el desarrollo.

4. Experimentación y evaluación del sistema desarrollado.
   Después de terminar el diseño e implementación, esta ha sido la fase más fácil y gratificante, puesto que me ha permitido probar el sistema de teleoperación, con cada una de sus características, y ver la sensación que causaba en las personas que lo experimentaban al realizar las pruebas.

5. Desarrollo de la documentación.
   Para llevar a cabo una buena descripción del trabajo desarrollado ha sido necesaria la documentación de cada una de sus fases, recogiendo así los detalles importantes. Permitiéndome a la vez, aprender algo más acerca de la historia y evolución de la robótica, y la correcta creación de un documento bien formateado.

6.3 Problemas encontrados

En este apartado se describen los problemas encontrados durante el diseño y el desarrollo de este trabajo.

- Durante el proceso de análisis de las distintas tecnologías utilizadas, encontré dificultades para poner instalar y configurar el entorno adecuado en el que se iba a desarrollar el sistema de teleoperación, debido a la falta de información existente, y los pocos ejemplos, habiendo poco información en Español, y muy pocas páginas en donde explicaran con claridad cómo utilizar el *software* proporcionado, a excepción de la página oficial de ROS.

- En el proceso de diseño se encontraron dificultades para definir qué sistema de teleoperación utilizar, ya que los movimientos realizados por el robot son complejos en algunos casos, por lo que hubo que dar muchas vueltas al diseño de los controles hasta encontrar con unos que se adaptaran bien al manejo del robot.

- En el desarrollo de la aplicación *NaoController* hubo dificultades para guardar correctamente el estado de la aplicación al producirse una interrupción en Android, como es el caso del giro de la pantalla, puesto que se destruye y se crea nuevamente la actividad y sus valores, si no se guardan correctamente, produciéndose errores en el sistema de teleoperación. Teniendo que guardar todos los valores relevantes continuamente para la correcta ejecución de la aplicación.

- La aplicación en Android tuvo que desarrollarse con varios hilos, ya que existen diversas partes que envían mensajes continuamente al nodo servidor, los mensajes de movimiento y los mensajes de video. Sin esto, al principio la aplicación solo mandaba mensajes de un tipo, pues solo una parte del programa se hacía con el control, impidiendo que la otra enviara mensajes.

- Para establecer el sistema de comunicaciones entre el nodo servidor y la aplicación *NaoController*, fue necesario el diseño de un estándar fijo en el paso de mensajes, y de esta forma conseguir que no hubiera problemas con los mensajes enviados y recibidos. Al inicio se producían errores en los mensajes, debido a la falta de coordinación entre ambos.

- El proceso de simulación para probar los movimientos de las distintas partes del cuerpo del robot, se llevó a cabo de manera satisfactoria. Pero se encontraron dificultades al intentar simular la visión del robot, ya que el entorno de simulación que se proporcionó, no contaba con esta opción; por lo que se tuvo que adaptar la cámara del portátil utilizado, por medio de un paquete para ROS, y así poder simular la captura de la imagen del robot, y transmitirla a la aplicación.

6.4 Líneas futuras

Este trabajo podría ser la base para la creación de un sistema de teleoperación más complejo para el robot Nao, donde se realicen acciones predeterminadas, o ser utilizado por otros robots. Se describen a continuación las posibles mejoras:

- Aumentar la velocidad de transmisión de los mensajes optimizando el código, ya sea en el módulo de control o en el módulo de teleoperación.

- Permitir la selección de la conexión a un robot de entre varios disponibles por medio de una lista.

- Permitir seleccionar al usuario de la aplicación que cámara tiene que utilizar el robot para visualizar el entorno, para poder realizar unas acciones u otras.

- Mejorar la precisión de los movimientos que se pueden realizar, por medio del uso de unos controles distintos o un sistema de reducción del error causado por el movimiento del dedo del usuario al utilizar la interfaz.

- Utilizar la aplicación como medio de comunicación con las personas con las que se encuentre el robot, a través de la escritura de mensajes en la aplicación y que esta se los envíe al robot y las reproduzca por medio de su sistema de voz.

- Utilización de la aplicación para el desarrollo de acciones preprogramadas concretas, como coger un objeto haciendo uso del sonar y su visión, subir escaleras, seguir a alguien por medio de la selección en la interfaz de la persona, recargarse si se observa que el nivel de batería es muy bajo…

- Servir como plataforma para utilizarla con otro tipo de robots, y poder seleccionar con qué tipo de robot se va a utilizar al iniciar la aplicación.

- Aumentar la información acerca del estado del robot, como por ejemplo el estado de las baterías, etc.

Glosario

Agentes físicos	Robots que realizan tareas por medio de la manipulación física del entorno en el que se encuentran.
Aibo	Mascota robótica diseñada y fabricada por Sony. Capaz de andar, ver su entorno y reconocer comandos hablados.
Android	Sistema operativo ligero diseñado primariamente para dispositivos móviles como *smartphones* y *tablets* que utilizaran procesadores ARM. Un objetivo secundario ha sido integrarlo en equipos de redes, *smart* TV, aparatos decodificadores e introducirlo en dispositivos como, relojes de pulsera.
API	Interfaz de programación de aplicaciones. Se le denomina al conjunto de funciones y/o procedimientos ofrecidos por una biblioteca, que pueden ser usados por otro *software* como capa de abstracción.
Array	(Matriz) disposición sistemática de objetos/datos, por lo general en filas y columnas.
Booleano	Tipo de dato lógico en computación que representa los valores de la lógica binaria, verdadero y falso.
Bumper	Sensor de presión utilizado en robótica para detectar golpes con objetos.
Bluetooth	Tecnología que posibilita la transmisión de datos entre distintos dispositivos mediante el uso de la radiofrecuencia en la banda ISM de los 2,4 GHz.
C++	Lenguaje de programación estáticamente escrito, multi-paradigma, compilado y de uso general. Es considerado un lenguaje e nivel medio, ya que tiene características de lenguajes de alto y bajo nivel. Desarrollado por Bjarne Stroustrup en 1979 en los laboratorios Bell. Agregó características orientadas a objetos, como clases y otras mejoras sobre el lenguaje C.
Checkbox	(Casilla de verificación) es un elemento gráfico de una interfaz de usuario que permite al usuario hacer

	selecciones múltiples a partir de una serie de opciones, o tener o no la confirmación del usuario sobre lo que el elemento se refiera.
Choregraphe	*Software* de programación diseñado y desarrollado por la empresa Aldebaran Robotics. Permite a los usuarios crear y editar movimientos y comportamientos interactivos para el robot NAO.
Ciclo de vida en espiral	Es un modelo de desarrollo de *software* formando una espiral, cada bucle representa un conjunto de actividades. Cada actividad se elige en función del análisis de riesgo de las anteriores.
Cinemática inversa	Es el uso de ecuaciones cinemáticas en un robot para permitir que uno de sus actuadores provisto de articulaciones se sitúe en una posición final concreta. Es un problema complejo que no tiene solución única.
Código abierto	*Software* que se desarrolla y distribuye libre.
Diagrama de casos de uso	Es un diagrama que representa comportamientos, con actores y casos de uso (acciones).
Diagrama de Gantt	Herramienta gráfica que se utiliza para mostrar el tiempo previsto de dedicación de una o varias actividades o tareas, a lo largo de un tiempo.
Dirección IP	Dirección de protocolo de internet, etiqueta asignada a cada dispositivo que participa en una red informática que utiliza el Protocolo de Internet para la comunicación. Tiene dos funciones principales: host o identificación de interfaz de red y dirección de localización.
Eclair	Versión 2.0 del sistema operativo Android.
Eclipse	Entorno multiplataforma de desarrollo de aplicaciones de código abierto.
Fedora	Distribución de Linux de código abierto para propósitos generales basada en RPM.
Framework	Es una plataforma universal y reusable usada para crear aplicaciones, productos y soluciones. Incluyen programas de apoyo, compiladores, librerías de código, interfaces de programación de aplicaciones (API's) y conjuntos de herramientas que reúnen juntas todos los componentes necesarios para permitir el desarrollo de un proyecto o solución.
Gingerbread	Versión 2.3 del sistema operativo Android.
Google docs	Es un sistema de almacenamiento de datos que provee un conjunto de herramientas que permiten a los usuarios crear y editar documentos en línea, permitiendo la colaboración en tiempo real.
Interfaz	Una herramienta y concepto que se refiere al punto de interacción entre componentes, y es aplicable en el nivel de *hardware* y *software*. Permite que los componentes de un sistema funcionen independientemente, y puedan comunicarse con otros componentes por medio de un sistema de entrada/salida y un protocolo asociado.

Java	Lenguaje de programación de alto nivel orientado a objetos. Su sintaxis deriva de C y C++ pero tiene un modelo de objetos más simple.
JDK	*Software* que proporciona herramientas para el desarrollo y la creación de programas en Java.
Kernel	Es el *software* más importante de los sistemas operativos. Provee acceso al *hardware* y gestiona los recursos.
Linux	Es un *kernel* de sistema operativo libre, que está basado en Unix.
Mac OS X	Sistema operativo creado por Apple, sucesor de Mac OS.
iOS	Antes iPhone OS, es un sistema operativo para móviles desarrollado y distribuido por Apple Inc. Originalmente lanzado en 2007 para el iPhone y el iPodTouch, se ha ampliado para soportar otros dispositivos de Apple como iPad y Apple TV.
LAN	Red de interconexión de ordenadores y periféricos.
Licencia BSD	Berkeley *Software* Distribution. Es una licencia de *software* libre, permitiendo la distribución.
Middleware	*Software* de ordenador que provee servicios a aplicaciones *software* más allá de los que están disponibles en el sistema operativo. Haciendo más fácil a los desarrolladores de *software* la comunicación de entrada/salida, pudiéndose concentrar así más en el propósito específico de su aplicación.
Multihilo	Capacidad para ejecutar múltiples hilos de ejecución dentro de las CPU's.
Multiplexar	Combinar varios canales de transmisión de datos a través de un solo medio.
NaoQi	Nombre del *software* principal que se ejecuta en el robot NAO y lo controla. El *framework* NaoQi, es el marco de programación usado para programar en el NAO. Responde a las necesidades comunes de la robótica como: paralelismo, recursos, sincronización, eventos…
Procesador ARM	Arquitectura de un procesador de 32 bits desarrollada por ARM Holdings.
Puerto	Interfaz por medio de la cual se envían y reciben datos.
Python	Lenguaje de programación de alto nivel, cuya filosofía es la legibilidad. Es fuertemente tipado, multiplataforma y soporta orientación a objetos.
SDK	Kit de desarrollo de *software*. Conjunto de herramientas que permite la creación de aplicaciones.
UAV`s	Vehículo aéreo no tripulado, comúnmente conocido como "drone", es un avión sin piloto humano a bordo. El vuelo está controlado de manera autónoma por computadoras en su interior o a distancia desde tierra u otro vehículo.
Ubuntu	Sistema operativo que funciona con un núcleo Linux, basado en Debian.
Unix	Sistema operativo multiusuario, portable y multitarea.
WAN	Red de computadoras de amplio alcance.

Referencias

[1] Emmanuel Nuño Ortega and Luis Basañez Villaluenga. (2004, Abril) http://upcommons.upc.edu/. [Online]. http://upcommons.upc.edu/e-prints/bitstream/2117/570/1/IOC-DT-P-2004-05.pdf

[2] A.M. Okamura, "Methods for haptic feedback in teleoperated robot-assisted surgery (Department of Mechanical Engineering, The Johns Hopkins University, Baltimore, Maryland, USA)," *Industrial Robot: An International Journal*, vol. 31, no. 6, pp. 499 - 508, 2004.

[3] Charles Baur, Terence Fong, and Charles Thorpe, "Advanced Interfaces for Vehicle Teleoperation: Collaborative Control, Sensor Fusion Displays, and Remote Driving Tools," *Autonomous Robots*, vol. 11, no. 1, pp. 77-85, 2011.

[4] Josas M. Sabater, Roque J. Saltarasn, Rafael Aracil, Eugenio Yime, and Josas M. Azorán, "Teleoperated parallel climbing robots in nuclear installations ," *Industrial Robot: An International Journal*, vol. 33, no. 5, pp. 381-386, 2006.

[5] R.R Murphy, "Trial by fire [rescue robots] (CRASAR, South Florida Univ., Tampa, FL, USA)," *Robotics & Automation Magazine, IEEE*, vol. 11, no. 3, pp. 50-61, Sept 2004.

[6] P. Boissy, D. Labonté, H. Corriveau, A. Grant.F. Michaud, P. Boissy, D. Labonté, H. Corriveau, A. Grant.F. Michaud, P. Boissy, D. Labonté, H. Corriveau, A. Grant. F. Michaud. (2007) Telepresence Robot for Home Care Assistance. [Online]. https://www.aaai.org/Papers/Symposia/Spring/2007/SS-07-07/SS07-07-012.pdf

[7] K. Nonami, "Intelligent Robots and Systems (Dept. of Electr. & Mech. Eng., Chiba Univ)," in *Proceedings. IEEE/RSJ International Conference*, 2000, pp. 775-779.

[8] Manuel Bogado Torres. (2007) http://oa.upm.es. [Online]. http://oa.upm.es/934/2/JUAN_MANUEL_BOGADO_TORRES.pdf

[9] Terrence Fong. Charles Thorpe. citeseerx.ist.psu.edu. [Online]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.1461

[10] AUROVA. (2008) http://www.disclab.ua.es. [Online]. http://www.disclab.ua.es/robolab/

[11] W., Fahrenholtz, J., and Leger, C. Amai. (2001) http://dl.acm.org/. [Online]. http://dl.acm.org/citation.cfm?id=591864

[12] E. Paulos. J. Channy. (2001) http://dl.acm.org/. [Online]. http://dl.acm.org/citation.cfm?id=591864

[13] University of Freiburg. Ros. [Online]. http://www.ros.org/news/2009/12/first-proper-release-of-freiburgs-nao-stack.html

[14] http://www.aldebaran-robotics.com. [Online]. http://www.aldebaran-robotics.com/en/Discover-NAO/Key-Features/hardware-platform.html

[15] http://www.willowgarage.com/.

[16] Morgan Quigley et al. http://www.robotics.stanford.edu. [Online]. http://www.robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf

[17] http://ros.org/. [Online]. http://ros.org/wiki/

Anexo A

Instalación y configuración del entorno

Para poder instalar el entorno adecuado se tiene que tener instalado Ubuntu como sistema operativo. Se recomienda instalar Ubuntu 10.04.

A.1 Instalación de ROS Diamondback

1. Configurar el repositorio de Ubuntu.

   Hay diversas opciones:

   - Sistema > Administración > Origenes del software.
   - Sistema > Administración > *Gestor de paquetes Synaptic* > Configuración > Repositorio.

     Y seleccionar las siguientes opciones:

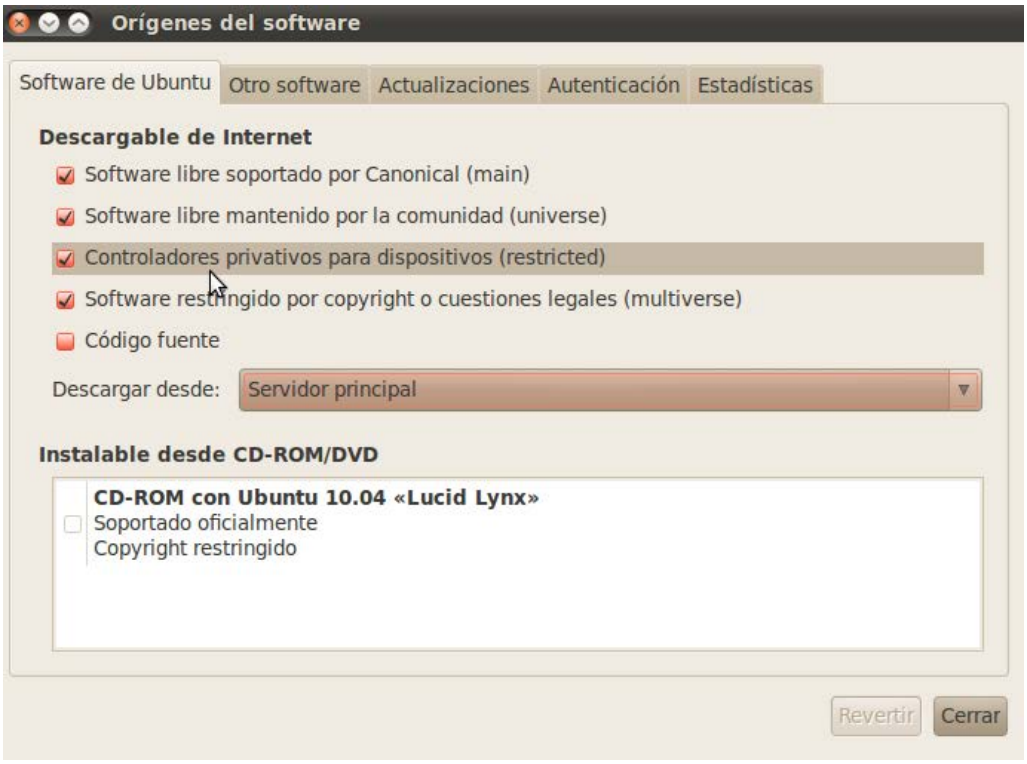

Figura 58. Orígenes del software Ubuntu.

A partir de aquí se abre un terminal y se escriben los comandos descritos.

2. Configurar el archivo sources.list para que el ordenador acepte software de Ros.org, mediante el siguiente comando:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu lucid main" > /etc/apt/sources.list.d/ros-latest.list'
```

3. Añadir las claves de los repositorios:

```
wget http://packages.ros.org/ros.key -O - | sudo apt-key add –
```

4. Instalación

   Para asegurar que se ha re-indexado el servidor ROS.org:

```
sudo apt-get update
```

   Se instala ros:

```
sudo apt-get install ros-diamondback-desktop-full
```

5. Configuración del entorno

   Es conveniente que las variables de entorno de ROS sean automaticamente

añadidas a la sesión bash cada vez que una nueva terminal sea lanzada:

```
echo "source /opt/ros/diamondback/setup.bash" >> ~/.bashrc
. ~/.bashrc
```

6. Configuración del fichero setup.sh

   Se navega hasta la carpeta donde se encuentra instalado ros y se modifica el fichero setup.sh:

```
cd opt/ros/diamondback/
```

```
sudo gedit setup.sh
```

   Dentro de este se copia lo siguiente:

```
#!/bin/sh

export ROS_ROOT=/opt/ros/diamondback/ros
export PATH=${ROS_ROOT}/bin:${PATH}
export NAOQI_PATH=/opt/naoqi/lib
export
PYTHONPATH=${ROS_ROOT}/core/roslib/src:${NAOQI_PATH}:${PYTHONPATH}
export ROS_PACKAGE_PATH=/opt/ros/diamondback/stacks
if [ ! "$ROS_MASTER_URI" ] ; then export
ROS_MASTER_URI=http://localhost:11311 ; fi
```

7. Configuración de permisos.

   Es conveniente dar permisos y hacerse propietario de todas las carpetas que se han creado para que luego no haya problemas de accesibilidad.

   Para modificar los permisos de lectura/escritura, se navega hasta la carpeta donde se encuentra ros y se escribe lo siguiente:

```
sudo chmod –R 777 ros
```

   En donde -R da permisos a todas las subcarpetas que hay por debajo y 777 da permisos de modificación.

   Para hacerse propietario de las carpetas se escribe:

```
sudo chown –R user:group ros
```

   En donde user es el nombre de usuario de la sesión de Ubuntu, y group es el grupo que haya en el equipo de trabajo.

A.2 Instalación y configuración del entorno ROS

1. Crear un workspace para ROS.

   Primero se crea el directorio del workspace, se crea en el directorio home.

```
mkdir ~/ros_workspace
```

   Ahora se crea un script bash para configurar el entorno ROS workspace. Se crea un archivo llamado setup.sh

```
sudo gedit setup.sh
```

   Y se le añade lo siguiente:

```
#!/bin/sh
 source /opt/ros/diamonback/setup.bash
 export ROS_ROOT=/opt/ros/diamondback/ros
 export PATH=$ROS_ROOT/bin:$PATH
 export PYTHONPATH=$ROS_ROOT/core/roslib/src:$PYTHONPATH
 export
ROS_PACKAGE_PATH=~/ros_workspace:/opt/ros/diamondback/stacks:$ROS_PACK
AGE_PATH
```

   Para añadir el ros_workspace al ROS_PACKAGE_PATH se escribe:

```
. setup.sh
```

   Para confirmar que la ruta ha sido añadida, se hace echo de la variable ROS_PACKAGE_PATH.

```
echo $ROS_PACKAGE_PATH
```

   Se debería ver algo como esto:

   /home/user/ros_workspace:/opt/ros/diamondback/stacks

   Para hacer este cambio permanente, asumiendo que el setup.sh está en el directorio home, se añade al final del archivo .bashrc, source ~/setup.sh de la siguiente manera:

```
sudo gedit .bashrc
```

   Dentro del archivo que se ha abierto se escribe lo siguiente:

```
source ~/setup.sh
```

2. Instalación de Naoqi y choregraphe.

   Se proveen en un cd el sdk de naoqi y choregraphe, ambos se deben extraer. Una vez extraidos, naoqi se debe de poner en la carpeta /opt/, se puede mover haciendo:

   ```
   sudo mv "origen" /opt/
   ```

   Donde origen es la ruta en la que se encuentra naoqi

   Puede surgir un problema sobre naoqi debido a la versión de python, se tiene que tener instalada la versión 2.6.2 de python.

   Se deben de descargar las librerías de geometry.msgs para que no den un error de import al utilizarlas.

   Se utiliza el siguiente comando para descargarlas:

   ```
   svn co "direccion del repositorio a descargar desde internet"
   ```

   Choregraphe no hace falta cambiarlo de sitio, se puede poner en cualquier carpeta.

3. Instalación del paquete de la universidad de Friburgo.

   Se descarga el paquete de la universidad de Friburgo accediendo a la siguiente dirección:

   http://code.google.com/p/alufr-ros-pkg/downloads/detail?name=nao_0.3.tar.gz&can=2&q=

   Una vez descargados, para instalar humanoid_navigation_ros, se hace como en este enlace.

   http://www.ros.org/wiki/humanoid_navigation

   Se descomprimen las dos carpetas que hay dentro (nao y humanoid_nav_msgs) y se mueven a /opt/ros/diamondback/stacks

   Se realiza:

   ```
   rosmake -rosdep-install humanoid_navigation
   ```

4. Instalación de eclipse.

Para instalar eclipse, accedemos a la página de eclipse y nos descargamos la última versión para java, la versión de 32 bits:

http://www.eclipse.org/downloads/

Se descomprime el archivo descargado, no hace falta instalación.

5. Instalación del JDK de java.

   Desde la terminal escribir:

```
sudo apt-get install openjdk-6jdk
```

6. Instalación del SDK de android.

   Se accede al siguiente link:

   http://developer.android.com/sdk/index.html

   Descarga la versión de android sdk para linux.
   Descomprimir el fichero descargado.

   Acceder a la carpeta resultante:

```
cd android-sdk-linux_86
```

   Ejecutar tools/android para descargar los paquetes que interesan. Se seleccionan todos, aunque no son necesarios y se instalan los seleccionados.

   Una vez se descargan, se accede a Eclipse desde el lugar donde se guardó, se selecciona Help → Install new software...

   Aparece una nueva ventana, se selecciona Add.. y aparece otra ventana, se añade el nombre y la localización del paquete para eclipse.

   Name: AndroidSDK
   Location: https://dl-ssl.google.com/android/eclipse/

   Se selecciona OK y aparece para elegir Developer tools, se selecciona, y se aceptan los términos de uso y se selecciona finish.

   Se reiniciará Eclipse, se selecciona Window → Preferences y en el menú lateral se selecciona Android. Donde pone SDK location, se selecciona Browse, y se busca la carpeta anteriormente descomprimida del SDK de Android, aplicamos con Apply y Ok.

   Para crear un simulador de Android, se selecciona Window → Android SDK and AVD Manager. Por último New…

Se escribe un nombre para la máquina virtual de android y se seleccionan los parámetros que se quieren tener de memoria y tamaño de pantalla.

Se selecciona Create AVD y ya se tiene creada la máquina virtual de Android.

Para iniciarla solo se tiene que seleccionar y darle a Start...

Anexo B

Diagrama de clases

En este apartado se describen los diagramas de clases del nodo servidor de ROS y la aplicación *NaoController*. Durante su desarrollo, ha sido necesaria la creación de diferentes clases que se comunican entre sí, de tal manera que con ellas se facilitarán la ampliación de posibles mejoras para los siguientes desarrolladores. Se muestran a continuación los diagramas de clases completos.

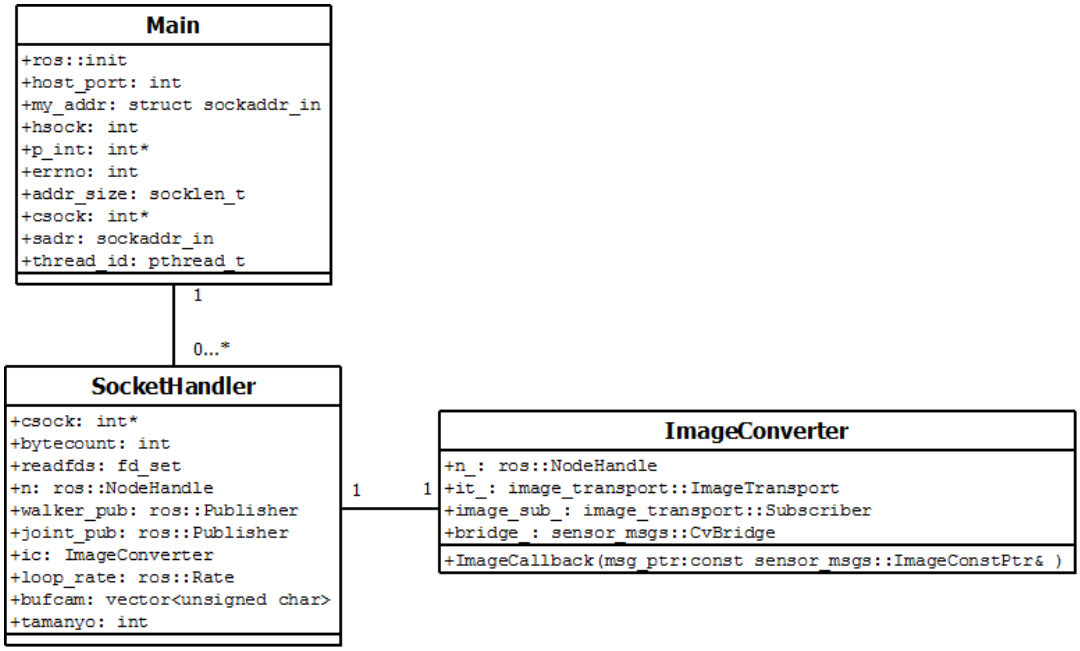

Figura 59. Diagrama de clases completo del nodo servidor.

A continuación se describen, de manera simple las clases de la Figura 59:

- ImageConverter: Esta clase se utiliza para obtener la imagen del robot, y transformarla a un formato específico que posteriormente la envíe a la aplicación *NaoController*.

- SocketHandler: En esta clase se lleva a cabo la suscripción y la publicación de mensajes al robot. Cuando una petición de movimiento o de imagen le llega al nodo servidor, esta clase compara que tipo de petición es, y en función de esto, publica el mensaje correspondiente o se suscribe a la información de interés.

- Main: En esta clase principal, se ejecuta el servidor y es en la que se espera a que un cliente se conecte, como es multihilo, se podrían conectar varios clientes a la vez.

A continuación se describen de manera simplificada, las clases plasmadas en la Figura 60:

- DualJoystickView: En esta clase se encuentran las variables y métodos que se usan para poder inicializar y tratar los *joysticks*.

- JoystickMovedListener: En esta clase se definen los métodos que se utilizan para ver en qué estado está el joystick, si está en movimiento, soltado o en la posición central.

- JoystickView: En esta clase se encuentran las variables y métodos que se usan para definir los *joysticks*.

- JoystickClickedListener: En esta clase se definen los métodos que sirven para ver si los *joysticks* se encuentran pulsados, o por el contrario se han soltado.

- NaoControllerActivity: En esta clase se encuentran las variables y métodos necesarios para utilizar la aplicación. Creación de la actividad principal, manejo y paso de mensajes, inicialización de los objetos que se utilizan en la interfaz y su posterior tratamiento.

- AppSocket: En esta clase se define el socket que se va a utilizar para la conexión de la aplicación, así como sus respectivos métodos para poder manejarlo. De esta manera la aplicación solo tiene un socket, y cuando se destruye la actividad principal debido a un giro de pantalla, o a un cambio de actividad, el *socket* se mantiene.

- HelpActivity: En esta clase se encuentra el método que crea la pantalla de ayuda de la aplicación.

- Help xml: En este xml se define como es la pantalla de ayuda de la aplicación.

- Main xml: En este xml se define como es la interfaz gráfica de la actividad principal de la aplicación.

- MainMenu xml: En este xml se define como es el submenú que aparece en la actividad principal al seleccionar el botón del dispositivo móvil de configuración.

- SettingsActivity: En esta clase está el método que crea el menú de ajustes de conexión de la aplicación.

- Preferences xml: En este xml se define como es el menú de ajustes de conexión de la aplicación.

Anexo C

Manual de uso de NaoController

En la Figura *61*, se muestra un diagrama de cómo se lleva a cabo la navegación en la aplicación *NaoController* por las distintas pantallas al utilizar la interfaz gráfica por el usuario. A continuación se explica brevemente como utilizar la aplicación creada utilizando como referencia esta figura.

1. Inicio de la aplicación: se pulsa el icono de la aplicación que aparece en el escritorio del dispositivo Android. A continuación se abrirá la interfaz de la aplicación.

2. Ajustes: se pulsa el botón del dispositivo móvil que despliega el menú de configuración, el botón no es el mismo en todos los dispositivos, depende del terminal. Aparece un menú con una serie de botones, entre ellos uno que se llama "Ajustes", se selecciona y aparecerá una nueva pantalla en donde se encuentran los ajustes para poder conectarse al servidor de ROS.

    - IP: se pulsa el campo IP para introducir la ip correspondiente al nodo servidor ROS. Aparece un teclado, se introduce el valor y se selecciona aceptar.

    - Puerto: se pulsa el campo Puerto para introducir el puerto en el que el servidor está escuchando. Aparece un teclado, se introduce el valor y se selecciona aceptar.

- Cámara de Nao: se selecciona este campo si se quieren ver las imágenes que el robot Nao captura por medio de una de sus cámaras.

3. Conexión con el servidor: se pulsa el botón del dispositivo móvil que despliega el menú de configuración. Aparece un menú con una serie de botones, entre ellos uno que se llama "Conectar", se selecciona y se espera a que se conecte con el servidor. Si este está ejecutándose y los parámetros de conexión se han introducido correctamente, la conexión se realizará satisfactoriamente. A partir de este momento, se puede empezar a controlar el robot. Si se quiere cancelar la conexión se realizan los mismos pasos descritos.

4. Ayuda: se pulsa el botón del dispositivo móvil que despliega el menú de configuración. Aparece un menú con una serie de botones, entre ellos uno que se llama "Ayuda", se selecciona y aparecerá una nueva pantalla en donde se encuentran las descripciones de cómo mover cada una de las partes del cuerpo del robot.

5. Salida: se pulsa el botón del dispositivo móvil que despliega el menú de configuración. Aparece un menú con una serie de botones, entre ellos uno que se llama "Salir", se selecciona y la aplicación se cerrará, acabando su ejecución y volviendo al escritorio del dispositivo Android.